ISSN 0868-6157
Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ



УСТРОЙСТВА ВВОДА ИНФОРМАЦИИ

9'91


demos /*

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

9'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

Адрес редакцнн:
113093, г.Москва, аб.ящнк 37
Факс: 200-22-89
Телефоны для справок:
491-01-53, 420-83-80.
E-mail:
postmaster@Computerpress.msk.su

Дорогой читатель!

С сегодняшнего дня мы начинаем принимать подписку на 1992 год. Бланк заказа помещен в конце этого номера. К сожалению, журнал опять подорожает, немного, но все-таки. Причины, наверное, можно не объяснять — по многочисленным требованиям трудящихся все дорожает. И этот процесс, похоже, необратим, по крайней мере, на ближайшие годы. Поэтому нас сегодня больше волнует, как не разочаровать читателя, потратившегося на подписку.

Что же изменится в следующем году? Во-первых, видимо, с обложки исчезнет подзаголовок "Обозрение зарубежной прессы", поскольку фактически уже давно журнал является обозрением не "зарубежной прессы", а зарубежного оборудования и программного обеспечения, причем, во многих случаях статьи пишутся чисто авторские, вообще без какой-либо связи с "зарубежной прессой".

Во-вторых, мы учтем упрек, высказанный нам читателем Иваном Тургеневым:

Мы за границу ездим, о друзья,
Как казаки в поход... Нам все не в диво;
Спешим, чужих презрительно браня,
Их сведений набраться торопливо...

То есть в следующем году мы будем больше уделять внимания отечественным разработкам и зарубежным продуктам, продающимся в стране, упорно бредущей по социалистическому пути. Действительно, какой смысл писать о винограде, ежели его не достать?

В-третьих, вышеупомянутые продукты будут нами тестироваться, а результаты тестирования будут публиковаться в новой рубрике, названне которой еще не придумано. Так что, если у кого появится желание сравнить свое детище, скажем, с Windows или с ScanMan 256, — милости проснм.

Сдано в набор 22.07.91. Подписано к печати 31.07.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). №032. Тираж 100 000 экз. (1 завод–55 000). Заказ 2399. Цена 3 р. 15 к.

Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18

Сегодня мы публикуем информацию, предоставленную нам Евгением Касперским. Важность ее оперативного распространения столь высока, что она будет одновременно опубликована в газете "СофтМаркет". Кроме того, мы разослали ее в сети RELCOM. Надеемся, что к моменту выхода этого номера в свет уже будет готов антивирус.

Внимание! Вирус "Driver-1024"

Атака очередного компьютерного вируса произошла в начале лета 1991 года. Одно из первых сообщений о заражении этим вирусом пришло из Львова. Оттуда он мигрировал в Киев, затем появился в Москве и Ленинграде (Санкт-Петербурге). Советские пользователи в основной своей массе уже не удивляются появлению вируса в их компьютере, однако тут им крупно "повезло" — они подверглись нападению вируса совершенно нового типа.

Основные отличительные черты вируса

1. Резидентен, поражает COM- и EXE-файлы. Длина вируса — 1024 байта.

2. При инициализации вирус проникает в ядро DOS, изменяет адрес системного драйвера дисков и затем перехватывает все обращения DOS к этому драйверу. В вирусе реализован мощный стелс-механизм на уровне системного драйвера, в результате чего вирус в зараженных файлах не виден при чтении файла как через int 21h, так и через int 25h. При этом вирус обращается напрямую к ресурсам DOS и "пробивает" практически любые антивирусные блокировщики.

3. Поражает логические диски, к которым происходит обращение DOS. Записывает свое тело в последний кластер инфицируемого диска. Этот кластер помечается как сбойный.

4. При заражении файлов их длины и содержимое кластеров, содержащих эти файлы, не изменяются. Вирус корректирует лишь номер первого кластера файла, расположенный в соответствующем секторе каталога. Новый начальный кластер файла будет указывать на кластер, содержащий тело вируса. Таким образом, на все зараженные файлы на одном логическом диске будет одна копия вируса!

5. Стремительно распространяется — поражает файлы при обращении DOS к секторам, содержащим каталоги. Например, при попытке запуска несуществующего файла DOS будет искать его во всех каталогах, отмеченных в PATH. При этом вирус перехватывает обращения DOS к каталогам и заражает файлы во всех каталогах, указанных в PATH. При первом старте вирус поражает все файлы текущего каталога диска C:.

Очень высокая скорость распространения, "невидимость" в файлах и, откровенно говоря, неготовность резидентных и нерезидентных антивирусных программ к появлению вируса такого класса делают этот вирус ОЧЕНЬ ОПАСНЫМ!

ОБНАРУЖЕНИЕ И ЛЕЧЕНИЕ

Обнаружение в памяти

Вирус содержит характерные участки кода, по которым можно произвести его поиск в оперативной памяти (участки кода приведены ниже). Поиск нужно производить в самом первом блоке памяти.

Квалифицированный пользователь может пользоваться другим способом: пройти список Drive Parameter Block (адрес первого DPB возвращается в таблице List of List при вызове int 21h, f.52) и искать драйверы, начинающиеся с адреса XXXX:04E9. По адресу XXXX:0100 будет располагаться начало вируса. Следует учитывать, что при прохождении списка заголовков драйверов вирус не будет обнаружен, так как он меняет лишь адрес драйвера в соответствующей DPB и не меняет коды и данные системных драйверов.

Обнаружение и лечение на диске

Для обнаружения вируса на диске лучше загрузиться с заведомо чистой от вирусов заклеенной дискеты, содержащей DOS и утилиты типа norton Disk Doctor и Norton Disk Editor. При тестировании диска NDD сообщит о том, что большое число файлов имеют общие участки (CROSS-LINKED), а последний кластер дис-

ка — сбойный. Этот сбойный кластер должен содержать тело вируса.

Лечение довольно просто производится при резидентном вирусе (то есть когда вирус инфицировал оперативную память). При этом достаточно переименовать все COM- и EXE-файлы в файлы с другими расширениями имени (например, в CCC- и EEE-файлы) или заархивировать их. Затем следует загрузиться с заведомо чистой от вирусов заклеенной дискеты и переименовать файлы обратно (или разархивировать их).

Если память компьютера не заражена вирусом, то лечение файлов является довольно непростой процедурой. Поэтому такие файлы лучше уничтожить.

Характерные участки вируса

```
0100  BC 00 06               MOV   SP,600H
0103  FF 06 EB 04            INC   [04EB]
0107  31 C9                  XOR   CX,CX
0109  8E D9                  MOV   DS,CX
010B  C5 06 C1 00            LDS   AX,DWORD PTR
DS:[00C1]
..........

; STRATEGY                        —>   CS:024B

024B  50                     PUSH  AX
024C  51                     PUSH  CX
024D  52                     PUSH  DX
024E  1E                     PUSH  DS
024F  56                     PUSH  SI
0250  57                     PUSH  DI
0251  06                     PUSH  ES
0252  1F                     POP   DS
0253  8A 47 02               MOV   AL,[BX + 2]

; INTERRUPT                  —>        CS:02A2

02A2  CB                     RETF

; VIRUS DEVICE HEADER        —>        CS:04E9

04E9  40                     INC   AX
04EA  C3                     RETN

04EB  xxxx                   DW    xxH
04ED  0842  Dev_Attr         DW    842H
04EF  024B  Dev_Strategy     DW    OFFSET STRATEGY
04F1  02A2  Dev_Intrrupt     DW    OFFSET INTERRUPT
```

Е.Касперский
109028 Москва, Тессинский пер., дом 6/19
Акционерное общество "КАМИ"
Научно-производственный отдел
телефон (8-095) 499-15-00

Lotus Development Corp. и The Santa Cruz Operation Inc. (SCO) пришли к согласию в судебном иске Lotus против SCO. Договорившиеся между собой, стороны решили, что Lotus снимает все свои судебные претензии к SCO, а последняя перестает производить и продавать свою программу работы с электронными таблицами SCO Professional.

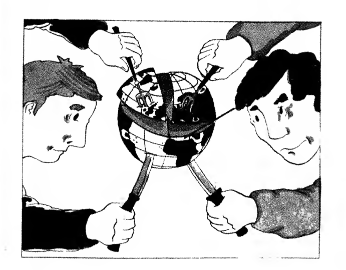
SCO также будет рекомендовать всем пользователям, уже купившим SCO Professional, перейти на Lotus 1-2-3 for UNIX System V, который поддерживает SCO UNIX System V/386, SCO Open Desktop и SCO XENIX 386. Чтобы поддержать эту рекомендацию, Lotus будет продавать этот пакет со скидкой для пользователей Santa Cruz — всего за 395 долларов.

Фирма Borland International, чья программа работы с таблицами Quattro Pro была также указана среди нарушителей авторского права в судебном иске Lotus, прокомментировала событие так: "Решение спора между SCO и Lotus не окажет никакого влияния на наш судебный процесс. Хотя мы и не можем предсказать его результаты, компания продолжает настаивать на том, что Quattro Pro является оригинальным продуктом и не нарушает никаких авторских прав на 1-2-3".

Ричард Шаффер, руководитель фирмы Technologic Partners, отметил следующее: "Я не думаю, что SCO испытывает сильное желание защищать свою табличную программу. Ее основной заботой является появление большего числа качественных прикладных программ, которые работают под SCO Unix. Поэтому, если Lotus действительно серьезно отнесется к своим обязательствам, исход процесса будет выгоден обеим сторонам. Хотя Borland и остается единственным противником Lotus в этом деле, его позиции остаются неизменными".

Финансовый аналитик, постоянно отслеживающий деятельность фирмы Lotus, сказал, что по его мнению процесс скоро получит новый толчок — Borland подала протест, заявляя, что фирма не имеет права требовать защиты авторских прав отдельно на систему меню пакета 1-2-3. "Отделение меню от остального продукта может иметь смысл в патентном процессе, а не при рассмотрении авторских прав".

Newsbytes News Network, 18 July, 1991



Все три сверхдержавы никогда не предпринимают маневров, чреватых риском тяжелого поражения. Если и осуществляется крупная операция, то, как правило, это внезапное нападение на союзника... Отсюда следует, что три державы не только не могут покорить одна другую, но и не получили бы от этого никакой выгоды. Напротив, покуда они враждуют, они подпирают друг друга подобно трем снопам.

Джордж Оруэлл, "1984", гл. IX

НОВЕЙШАЯ ИСТОРИЯ КОМПЬЮТЕРНЫХ ВОЙН: кошмары по Оруэллу

С апреля этого года читатели массовой прессы вынуждены осваивать язык арифметических выражений. "9+1", "15+0", "9+6+X". Следуя этому стилю, апрельскую новость, о которой вам рассказал КомпьютерПресс №5, можно записать с помощью оператора присваивания ACE = 3+17. Среди трех лидеров компьютерного мира, возглавивших работы по стратегической программе ACE, — фирма Microsoft. С тех пор минуло всего лишь три месяца, и вот...

Посещают ли кошмары сон Билла Гейтса? Недавно он сам признался — посещают. И даже рассказал, какие. "И вот снится мне, что IBM нас атакует с системного фланга, Novell — с локально-сетевого, а любители оконных интерфейсов с лозунгом на стяге ОКОШКИ НАШИ ЛУЧШЕ ВАШИХ объединились и напирают по всему фронту. А самое кошмарное — то, что это так и есть".

Смягчить впечатление от сказанного Гейтсом на июньской оперативке администрации фирмы Microsoft попытался его коллега Стив Болмер. Он философски заметил, что удел бизнесмена — всегда быть готовым к худшему. В том числе и к падению биржевого курса акций фирмы в конце июня. И к печальному для Microsoft развитию событий в расследовании, проводимом Федеральной комиссией по делам торговли, которая интересуется возможными нарушениями антимонопольного законодательства со стороны Microsoft. Да еще Фирма Apple Computer через своих адвокатов нетактично "подбрасывает" вопрос о правах Microsoft на применение в своих продуктах метафоры оконного интерфейса.

Все эти, говоря военно-полевым языком, вводные обрушились на фирму Microsoft не вчера. И ответные действия ее показали, что с порохом в пороховницах проблем нет. Апрельское коммюнике группы фирм, объявивших о развертывании стратегической программы ACE (см. КомпьютерПресс №5 за этот год), свидетельствует о присущем членам этого альянса чувстве перспективы. Действительно, объединить в общей операционной среде функции MS DOS, OS/2, MS Windows и ядра POSIX для массовых персональных компьютеров начала 90-х годов — цель прагматичная и отвечающая сложившемуся у пользователей имиджу фирмы Гейтса. Мол, Гейтс — это опытный сталкер, который выведет в опасной зоне каждого клиента к причитающемуся оному клиенту счастью. Счастье может прийти нежданно, если не боишься рискнуть, — и RISCуют, отправившись за Гейтсом со товарищи в зону, где их ждут рабочие станции с микропроцессором MIPS R4000 внутри.

А тем временем "синий гигант" решил, что ему не помешают лавры Синей Бороды. Многолетний брак фирм IBM и Microsoft, принесший миру обильное потомство, оказался перед перспективой стремительного

разрыва отношений. Кого выберет в партнеры IBM? Оказалось, что в амурах этому персонажу нет равных: такого никто не ожидал. Новый альянс свел под венцом таких конкурентов, для которых любое другое партнерство было бы лишь скучной регистрацией легкой победы. Мало ли фирм и фирмочек побывало в доме IBM на правах ветреных служанок... Здесь же все серьезно: фирма Apple Computer вступает в законный брак с IBM, и уже имя первенцу придумали. "Совместно созданная и независимо функционирующая фирма для разработки мобильных объектно-ориентированных программных средств". Длинновато, но запоминается.

Итак, 3 июля этого богатого событиями года фирмы IBM Corporation и Apple Computer Inc. подписали протокол о намерениях, обусловленный заключением в ближайшее время конкретных соглашений. Среди намеченных направлений работ — поставка программных средств, создаваемых совместным предприятием IBM-Apple, на рынок персональных компьютеров (в том числе и для "третьих поставщиков"). Какие же платформы будут годиться для этих мобильных программ? Любые, если они содержат архитектурное ядро "X86 ИЛИ 680X0 ИЛИ RISC POWER". О последнем терме в этом выражении надо сказать отдельно. RISC-процессор, разработанный инженерами IBM для мощных рабочих станций, дороговат для массовых компьютеров. И это обстоятельство заставило фирму IBM сделать еще один важный шаг — альянс становится с самого начала тройственным. Фирма Motorola, у которой есть давний опыт сотрудничества с IBM (вместе пытались разместить в наборе БИС архитектуру IBM System/370), вместе с IBM выпустит скоро дешевую версию БИС для архитектурного ядра POWER. На судьбе родного детища Motorola 88000 этот выбор может сказаться роковым образом.

Выбор весьма популярной сегодня объектно-ориентированной технологии программирования в качестве общей идеологии также выглядит естественным. IBM получает доступ к богатому опыту Apple в этой области, и вместе они намерены доработать систему AIX таким образом, чтобы в ней поддерживались графические пользовательские интерфейсы Macintosh и OSF/Motif. Конечная цель здесь — добиться, чтобы прикладные программы для трех операционных систем предыдущего поколения — OS/2 (теперь это приемный ребенок IBM?..), AIX и System 7.0 Macintosh — работали в общей операционной среде.

Встречный интерес фирмы Apple направлен прежде всего к опыту IBM в области сетевых архитектур. Здесь брезжит решение ключевой для Apple проблемы: доступа к крупным заказам на комплексную автоматизацию предприятий, где часто побеждало стремление заказчика иметь дело с одним поставщиком по спектру от больших машин до терминалов. Интересно вспомнить, что альянс на эту же тему был оформлен всего пару лет назад между Apple и Digital. Какая его ждет теперь судьба?

Наконец, особо оговорено, что Apple и IBM отдадут высокий приоритет совместным разработкам в области информационной технологии Multimedia.

На развертывание всех упомянутых выше работ фирмы дали своим инженерам не более двух-трех лет.

В интервью корреспонденту информационного агентства NEWSBYTES представительница Apple Барбара Краузе заявила: "Это соглашение позволит нам продвинуться в достижении трех важных целей. Во-первых, мы наконец всерьез приступим к развитию RISC-платформы для ряда моделей Macintosh. Во-вторых, приобретение имиджа в RISC-технологии поможет нам продвинуться на рынок больших корпораций, к чему мы всегда стремились. В-третьих, интеграция с высокопроизводительными компьютерами была и остается для нас ключевой проблемой".

Пока что слухи о кандидатуре руководителя для создаваемого совместного предприятия IBM-Apple Краузе ни подтвердила, ни опровергла. Им может стать Дейвид Лиддл — глава динамичной молодой фирмы Metaphor Systems.

Наши цели ясны, задачи определены — за работу, господа? Нас ожидает борьба двух действительно могучих триумвиратов — IBM/Apple/Motorola и Microsoft/Compaq/Digital в наиболее сложном секторе компьютерного рынка. И хотя ни один из шести партнеров не может быть заподозрен в желании таскать из огня каштаны для кого-то еще, в ожидании жареных каштанов уютно устроились две сторонящиеся этих игр фирмы. Это — всего-навсего AT&T и Intel.

В эти жаркие июльские дни усталый вздох вырвался из груди ветерана компьютерной журналистики Джона МакКормика (агентство NEWSBYTES): "Теперь можно и на пенсию... Могу смело сказать — в этой индустрии я повидал все, что только может случиться."

А случаи — что апрельский, что июльский — и впрямь судьбоносные. Джордж Оруэлл не отказался бы включить эти сюжеты в продолжение своего "Тысяча девятьсот восемьдесят четвертого": сверхдержавы молниеносно перестраивают на карте мира свои боевые колонны, а идеологи находят этому единственно верное обоснование. Враги? Союзники? Все это преходяще, и вообще, ведь именно программисты первые отметили, что любая константа — всего лишь очень медленно изменяющаяся переменная... "Пожалуй, скоро герою войны в Заливе генералу Шварцкопфу предложат должность председателя правления в компьютерной фирме, где исполнительным директором — Дейвид Лиддл", — комментирует происходящее Джон МакКормик.

А.Гиглавый (Лицей информационных технологий),
К.Чащин (московское отделение информационного агентства NEWSBYTES)

Суета вокруг Роберта или Моррис-сын и все, все, все...

...Компьютерный мир развивался. И вот в 1969 году в США появилась глобальная сеть Arpanet, соединившая тысячи компьютеров и огромное количество пользователей. Возможность работы с коллегой, находящимся за тысячу километров, привлекла многих. В сети работали сотрудники и студенты унивеситетов и колледжей, подрядчики Пентагона. В том же 1969 году появилась на свет операционная система UNIX, быстро завоевавшая широкую популярность. И, казалось, ничто не предвещало бури...

Итак, 1988 год. Лето. В AT&T Bell Laboratories работает студент выпускного курса одного из университетов. Он занимается программами обеспечения безопасности UNIX. Зовут молодого человека Роберт Моррис.

2 ноября 1988 года. Вечером в разных концах страны на компьютерах, работающих под UNIX, был обнаружен вирус. Постепенно выяснилось, что вирус одинаков на всех пораженных машинах. В результате его работы загрузка систем повышалась, а после достижения ею критического уровня системы переставали работать.

Вирус распространялся с поразительной скоростью и появлялся в разных концах страны. Был сделан вывод, что вирус распространяется через сеть. Через 5 часов было поражено около 800 систем, через двое суток — 6000.

Атака велась на самые популярные сети, в том числе пентагоновские, и на самую популярную операционную систему. К мероприятиям по обезвреживанию вируса подключились представители ФБР. Уже 4 ноября они нашли файлы, использованные при создании вируса. Их владельцем оказался Роберт Таппан Моррис. В тот же день он явился с повинной в штаб-квартиру ФБР в Вашингтоне...

Что за этим последовало

Воскресенье, 6 ноября 1988 года.

15:50 RISKS Высказаны предупреждения против раздувания большей шумихи вокруг вируса, чем это сделано в прессе. Событие, "происшедшее в действительности, не является сюрпризом ни для читателей RISKS, ни для пользователей Internet или UNIX".

19:01 RISKS Краткое обращение к пользователям с призывом изучать вирус; отмечается, что "неэтичность и другие злоупотребления не являются необычными". Компьютерные системы должны перестать быть "относительно широко открытыми".

Понедельник, 7 ноября 1988 года.

15:44 RISKS Обсуждение возможности провокации вирусом ядерной войны. Вызвано воскресным сообщением (15:50), но впадает в противоположную крайность, преувеличивая возможность "ядерного Армагеддона".

19:06 VIR Частная реакция на вирус Internet компьютерной службы, которая "непосредственно даже не связана с Internet" и лишь к этому моменту была оповещена из местных газет и местных филиалов ABC и NBC.

8 ноября 1988 года в Вашингтоне была организована встреча группы программистов и экспертов по компьютерной безопасности из различных университетов с одной стороны и представителей федеральных властей с другой — "Proceedings of the Virus Post-Mortem Meeting".

Целью встречи было обсуждение результатов вирусной атаки и сложившейся в связи с этим ситуации и выработка предложений относительно того, как избежать повторения подобных атак в будущем. На встрече был принят документ, состоящий из 11 рекомендаций участников совещания.

Одним из результатов встречи стал запрос NCSC к исследователям университета в Пурду на удаление из компьютерной системы университета всей информации, касающейся алгоритма работы вируса. Университет удалил всю информацию, которая, с точки зрения специалистов, представляла какую-либо опасность.

Согласно рекомендациям была создана так называемая Группа ответа на компьютерную опасность

Окончание. Начало см. КомпьютерПресс №8, 1991.

(Computer Emergency Response Team). В настоящее время группа ответа официально располагается в университете Карнеги-Меллона, находясь на финансовом обеспечении серьезно обеспокоенного инцидентом Пентагона.

РЕКОМЕНДАЦИИ СОВЕЩАНИЯ, СОСТОЯВШЕГОСЯ 8 НОЯБРЯ 1988 ГОДА

I. Образовать единый координационный центр на случай подобных атак.
Центр должен работать совместно с NIST и NSA, выполняя функции обобщения данных и определения ущерба.

II. Образовать сеть аварийного оповещения.
Такой аварийной сетью связи может служить совокупность специальных телефонных линий, которая сможет заменить компьютерную сеть в случае выхода последней из строя по каким-либо причинам.

III. Создать группу ответа.
Члены группы должны иметь целью быстрое дезассемблирование вируса, создание антивируса и выработку плана восстановления сети.

IV. Образовать физическую связь с "сетью стариков" <old boy (sic) network> — признанных авторитетов в области информатики и вычислительной техники.
Это является косвенным признанием того, что вирус Морриса был дезассемблирован и уничтожен не правительственными чиновниками, а специалистами-компьютерщиками.

V. Централизованно управлять предоставлением информации прессе.
Связь с прессой — единственная обязанность правительственных органов национального уровня.

VI. Определить стандартную процедуру реакции на "промышленные промахи."
Данная атака выявила целый ряд ошибок в системном программном обеспечеиии, однако на данный момент не существует общепризнанного метода доказательства "промышленного" характера этих ошибок.

VII. Определить центральное место размещения сообщений о вирусных атаках.
Эти функции может выполнять EBB Dockmaster правительственной компьютерной сети.

VIII. Подключать следственные органы уже на этапе планирования и проведения мероприятий борьбы с вирусной атакой.
Это позволит следственным органам своевременно фиксировать информацию, которая впоследствии будет служить основой для расследования и формулировки обвинений.

IX. Постоянно тренировать операторов систем.
Вирусная атака показала, что многие операторы не имеют достаточных технических знаний и практических навыков даже для понимания, что их система атакована. Они также с трудом обращаются с антивирусными средствами.

X. Установить стандартную методологию backup-копирования.
Существующая система "зеркального" копирования потерпела крах при испытании вирусной атакой, поэтому должны быть разработаны новые стандарты и критерии копирования, информация о которых должна распространяться NIST и NCSC.

XI. Развивать набор общих антивирусных средств, включая средства дезассемблирования и анализа.
Этот набор общих средств необходимо постоянно пополнять и поддерживать в работоспособном состоянии, а также необходимо обеспечить надежный и быстрый доступ к этому набору членов группы ответа.

ФБР наложило запрет на все материалы, имеющие отношение к вирусу Морриса, а 11 ноября 1988 года в "Нью-Йорк Таймс" в статье Джона Маркоффа было заявлено, что NCSC ищет способы, чтобы вообще остановить распространение точной информации о "внутренней работе программ, которые осложняют работу американских компьютеров".

Теперь, я думаю, вам стало понятно, почёму получить подробные материалы (и по вирусу Морриса, в частности) в настоящее время не так-то просто.

Вирусная атака послужила причиной резко возросшего интереса к средствам обеспечения безопасности как вычислительных систем, так и объединяющих эти системы сетей.

Правительственные лаборатории и исследовательские центры, занимающиеся проблемами защиты информации, интенсифицировали работы над созданием сложных и, по возможности, универсальных антивирусных программ, которые должны выявлять и блокировать вирус на самых ранних стадиях его развития. Однако, по мнению экспертов, если такие программы и удастся создать, они вряд ли станут достоянием гласности и, вероятнее всего, в случае создания универсальных антивирусных программ они будут рассматриваться как своего рода "стратегическое оружие".

К работам по борьбе с вирусами и созданию "иммунных" программ подключились Национальный научный фонд США, биржа Уолл-Стрита и др. Беспокойство финансовых кругов США объясняется их сильной зависимостью от вычислительной техники, на базе которой функционирует вся система электронных безналичных расчетов.

Промышленная ассоциация по компьютерным вирусам только за 1988 год зафиксировала почти 90 тысяч вирусных атак на персональные компьютеры, а по данным Национального центра информации по компьютерной преступности (National Center for Computer Crime Data), за этот же год компьютерная преступность нанесла американским фирмам убытки в размере 500 млн. долларов. В наибольшей степени от этого страдают банки. По словам Ховарда Глассмана, который отвечает за безопасность в Bank of America, компьютерная преступность может стать причиной крушения экономической системы страны.

На самом деле, количество инцидентов, связанных с вирусами, вероятно, превосходит опубликованные цифры, поскольку большинство фирм умалчивает о вирусных атаках, опасаясь, что подобная "реклама" повредит их репутации. Другие фирмы молчат, чтобы не привлекать внимание хакеров. Как заметил администратор фирмы Amway, "лучший способ избежать проблем с вирусами — не болтать о том, что вы предпринимаете во избежание этих проблем".

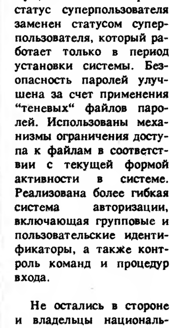


Поскольку вирус Морриса ощутимо ударил по репутации одного из популярнейших семейств операционных систем — UNIX, совсем неудивительно, что разработчики стали прилагать лихорадочные усилия восстановления былого авторитета своего детища и соответственно для удержания рынка сбыта.



Так, специалистов калифорнийского университета в Беркли — места рождения UNIX Berkeley — вирусная атака побудила заново полностью проанализировать свою систему, исправить обнаруженные червем дыры и отключить в системе UNIX режим отладки. Они также переписали программу Finger, научив ее проверять границы размещаемой ею памяти с тем, чтобы любой проникший в систему вирус не смог бы записать в эту память свои коды.

Как сообщил в феврале 1989 года вице-президент компании AT&T Вильям О'Ши (William O'Shea), для системы UNIX 4.1 были разработаны улучшенные средства обеспечения безопасности. Новшества состояли в усовершенствовании методологии доступа, обеспечении целостности данных, отказе от утилит и проведении ненавязчивой политики сдерживания. Существовавший ранее статус суперпользователя заменен статусом суперпользователя, который работает только в период установки системы. Безопасность паролей улучшена за счет применения "теневых" файлов паролей. Использованы механизмы ограничения доступа к файлам в соответствии с текущей формой активности в системе. Реализована более гибкая система авторизации, включающая групповые и пользовательские идентификаторы, а также контроль команд и процедур входа.



Не остались в стороне и владельцы национальных компьютерных сетей, в частности Internet. Internet Activities Board (комитет Internet) в феврале 1989 года заявил о намерении ввести с целью увеличения сохранности сообщений электронной почты в Internet новый стандарт безопасности. Пользователи получат возможность шифровать свои сообщения, а также проверять идентичность отправленного и полученного сообщений. Стандарт безопасности включает DES, систему аутентификации из RSA Data Security и формат аутентификации CCITT X.509.

ПАРОЛИ, ОПРОБОВАННЫЕ ВИРУСОМ МОРРИСА

aaa	*academia*	*aerobics*	*airplanealbany*	*albert*	*alex*	*alexander*	*algebra*	*aliases*
alphabet	*ama*	*amorphous*	*analog*	*anchor*	*andromache*	*animals*	*answer*	
anthropogenic	*anvils*	*anything*	*aria*	*ariadne*	*arrow*	*arthur*	*athena*	*atmosphere*
azure	*bacchus*	*bailey*	*banana*	*bananas*	*bandit*	*banks*	*barber*	*baritone*
bass	*bassoon*	*batman*	*beater*	*beauty*	*beethoven*	*beloved*	*benz*	*beowolf*
berkeley	*berliner*	*beryl*	*beverly*	*bicameral*	*bob*	*brenda*	*brian*	*bridget*
broadway	*bumbling*	*burgess*	*cager*	*campanile*	*cantor*	*cardinal*	*carmen*	*caroline*
castle	*cascades*	*cat*	*cayuga*	*celtics*	*cerulean*	*change*	*charming*	*charles*
charon	*chester*	*cigar*	*classis*	*coffee*	*computer*	*condo*	*cookie*	*cornelius*
couscous	*creation*	*creosote*	*cretin*	*daemon*	*dancer*	*daniel*	*danny*	*dave*
december	*defoe*	*deluge*	*desperate*	*develop*	*dieter*	*digital*	*discovery*	*disney*
dog	*drought*	*duncan*	*eager*	*easier*	*edges*	*edinburgh*	*edges*	*edwin*
edwina	*egghead*	*eiderdown*	*eileen*	*einstein*	*elephant*	*elizabeth*	*ellen*	*emerald*
engine	*engineer*	*enterprise*	*enzyme*	*ersatz*	*establish*	*estate*	*euclid*	*evelyn*
extension	*fairway*	*felicia*	*fender*	*fermat*	*fidelity*	*finite*	*fishers*	*flakes*
float	*flower*	*flowers*	*foolproof*	*football*	*foresight*	*format*	*forsythe*	*fourier*
fred	*friend*	*frighten*	*fun*	*fungible*	*gabriel*	*gardner*	*garfield*	*gauss*
george	*gertrude*	*ginger*	*glacier*	*gnu*	*golfer*	*gouge*	*graham*	*guest*
guitar	*guntis*	*hacker*	*hamlet*	*handily*	*happening*	*harmony*	*harold*	*harvey*
hebrides	*heinlein*	*hello*	*help*	*herbert*	*hiawatha*	*hibernia*	*honey*	*horse*
horus	*hutchins*	*imbroglio*	*imperial*	*include*	*ingres*	*inna*	*innocuous*	*irishman*
isis	*japan*	*jessica*	*jester*	*jixian*	*johnny*	*joseph*	*joshua*	*judith*
juggle	*julia*	*kathleen*	*kermit*	*kernel*	*kirkland*	*knight*	*ladle*	*lambda*
lamination	*larkin*	*larry*	*lazarus*	*lebesque*	*lee*	*leland*	*leroy*	*lewis*
light	*lisa*	*louis*	*lynne*	*macintosh*	*mack*	*maggot*	*magic*	*malcolm*
mark	*markus*	*marty*	*marvin*	*marty*	*master*	*maurice*	*mellon*	*merlin*
mets	*michael*	*michelle*	*mike*	*minimum*	*minsky*	*moguls*	*moose*	*morley*
mozart	*napoleon*	*ness*	*network*	*newton*	*next*	*noxious*	*nutrition*	*nyquist*
oceanography	*ocelot*	*olivetti*	*olivia*	*oracle*	*orca*	*orwell*	*osiris*	*outlaw*
oxford	*pacific*	*painless*	*pakistan*	*pam*	*papers*	*password*	*patricia*	*peoria*
penguin	*percolate*	*persimmon*	*persona*	*pete*	*peter*	*philip*	*phoenix*	*pierre*
pizza	*plover*	*plymouth*	*polynomial*	*pondering*	*pork*	*poster*	*praise*	*precious*
prelude	*prince*	*princeton*	*protect*	*protozoa*	*pumpkin*	*puneet*	*puppet*	*rabbit*
rachmaninoff	*rainbow*	*raindrop*	*raleigh*	*random*	*rascal*	*really*	*rebecca*	*remote*
rick	*ripple*	*robotics*	*rochester*	*rolex*	*romano*	*ronald*	*rosebud*	*rosemary*
roses	*ruben*	*rules*	*ruth*	*sal*	*saxon*	*scamper*	*scheme*	*scott*
scotty	*secret*	*sharks*	*shuttle*	*signature*	*simon*	*simple*	*singer*	*single*
smile	*smiles*	*smooch*	*smother*	*snatch*	*snoopy*	*soap*	*socrates*	*sossina*
sparrows	*spit*	*spring*	*springer*	*squires*	*strangle*	*statford*	*stuttgart*	*subway*
success	*summer*	*super*	*superstage*	*support*	*supported*	*surfer*	*suzanne*	*swearer*
symmetry	*tangerine*	*tape*	*target*	*tarragon*	*taylor*	*telephone*	*temptation*	*thailand*
tiger	*toggle*	*tomato*	*topography*	*tortoise*	*toyota*	*trails*	*trombone*	*trivial*
tubas	*tuttle*	*umesh*	*unhappy*	*unicorn*	*unknown*	*urchin*	*utility*	*vasant*
vertigo	*vicky*	*village*	*virginia*	*warren*	*water*	*weenie*	*whatnot*	*whiting*
whitney	*will*	*william*	*williamsburg*	*willie*	*winston*	*wisconsin*	*wizard*	*wombat*
yellowstone	*yosemite*	*zap*						

КОЛЛЕГА!

При выборе своего пароля избегай использования имен жены или любовницы и вообще имен, не используй названия городов, стран и вообще названий! Для тех, кто не в состоянии придумать сам какую-нибудь абракадабру, советую при выборе пароля хотя бы сверяться с приведенным списком, чтобы не уподобляться десятому слову четвертого столбца таблицы.

Защищать сети гораздо сложнее, чем системы. Так, например, в Internet защите подлежат три основных точки: индивидуальные терминалы, локальные сети и собственно сеть Internet, которая может быть поражена из любой сети, из числа объединяемых ею. Большое значение в связи с этим, помимо аппаратно-программных средств, приобретают организационные меры, предпринимаемые в целях обеспечения безопасности сети вплоть до контроля за дисциплинированностью каждого пользователя.

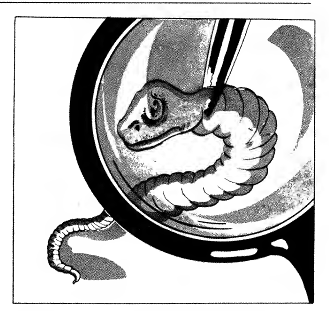
Для защиты Internet специалистами, в частности, было предложено следующее:

- должна быть сформирована политика безопасности сети, включая вопросы уголовного преследования нарушителей;
- операторы Internet должны начинать уголовное преследование всех лиц, замешанных в атаке сети;
- операторы Internet должны начать регулярные тренировки с упором на обеспечение безопасности сети;
- Internet должна оплачивать информацию, помогающую обнаружить и доказать виновность лиц, атаковавших сеть.

После достопамятной атаки пользователи стали обвинять во всем случившемся не только Морриса, но и транспортный протокол TCP/IP, утверждая, что сам

протокол не соответствует требованиям обеспечения безопасности. Однако в ответ специалистами было заявлено, что вирус вызвал серьезные последствия в силу недостатков средств обеспечения безопасности систем, но не сети.

Вирус распространялся через программы, входящие в систему UNIX, которые, как оказалось, были написаны недостаточно хорошо и имели дыры в схеме безопасности. Пикантность ситуации состояла в том, что существуют другие программы, работающие по тем же алгоритмам ничуть не хуже, а то и лучше упомянутых злосчастных программ, но при этом не имеющие дыр в безопасности. Те системы в Internet, которые использовали эти альтернативные программы, избежали поражения вирусом. Это доказывает, что TCP/IP в состоянии поддерживать безопасный трафик.



Как это судили

"НЬЮ-ЙОРК, 9. (ТАСС) Пока Роберт Моррис лишь смущенно улыбается перед объективами репортеров — он уже стал героем дня в США, но еще не ясно, смогут ли американские власти доказать, что он еще и преступник. "У нас нет никакого опыта наказаний за такие дела", — заявил представитель ФБР..."

Советская Россия, 10.11.88

"Вся эта история выглядит и как захватывающая драма отца, сына и колоссальной электронной игры, и как наихудший результат веселой проделки, в которой непреднамеренность не может служить оправданием. Разбор этого дела в суде позволит выяснить соответствие нынешнего законодательства, направленного против "электронного саботажа", уровню развития вычислительных систем и средств. Очень важно доказать преступность подобных действий, хотя в данном конкретном случае безусловно имеет смысл смягчить наказание".

Х.Д.Хайланд

Четверг, 10 ноября 1988 года.

19:03 VIR Утверждение, что "действия, предпринятые в отношении" Морриса и подобных ему лиц будут "действиями общества в области компьютерной безопасности".

20:40 RISKS Официальное сообщение о ситуации в Корнеллском университете, откуда, по-видимому, был запущен вирус. Выводы сопровождаются комментарием, какой Моррис хороший; высказывается уверенность, что его наказание не будет "слишком суровым".

Представляет интерес юридическая сторона дела.

Долгое время ФБР не могло выдвинуть официального обвинения в отношении автора вируса, хотя по заказу этого уважаемого ведомства Гарвардским и Корнеллским университетами были подготовлены обзоры существующего законодательства, касающегося "компьютерных" преступлений. Несмотря на наличие относительно новых законодательных актов, таких как Закон о злоупотреблениях и мошенничестве с помощью компьютеров от 1986 года (Computer Fraud and Abuse Act of 1986), выдвинуть обвинение в отношении лица, злоупотребившего многомашинной компьютерной системой, каковой является любая компьютерная сеть, было нелегко.

В конце концов, поскольку пораженные вирусом системы входили в состав компьютерной сети, контролируемой правительством США, распространение вируса было признано действием, нарушившим закон. Но и после этого осталось неясным, как квалифицировать действия автора вируса — как преступление или как обычное хулиганство. В соответствии с уже упоминавшимся законом от 1986 года в случае квалификации действий Морриса как "несанкционированного доступа к правительственным компьютерам", ему грозил крупный штраф и тюремное заключение сроком до 10 лет.

22 января 1989 года суд присяжных признал Морриса виновным. Если бы осуждающий вердикт был утвержден без изменений, то Морриса ожидали бы 5 лет тюремного заключения и 250000 долларов штрафа. Однако адвокат Морриса Томас Гидобони (Thomas Guidoboni) сразу же заявил протест и направил все бумаги в окружной суд с прошением отклонить решение суда.

Защита Морриса основывалась на двух посылках:

- на том, что формулировка закона неопределенна и

СКОЛЬКО СТОИЛ ВИРУС МОРРИСА

По самым скромным оценкам, инцидент с вирусом Морриса стоил свыше 8 миллионов часов потери доступа и свыше миллиона часов прямых потерь на восстановление работоспособности систем. Общая стоимость этих затрат оценивается в 98 миллионов долларов. Ущерб был бы гораздо большим, если бы вирус изначально создавался с разрушительными целями.

Столь необычайно большая сумма ущерба объясняется гигантскими масштабами пораженных сетей (в основном — Internet) и значительным количеством пораженных систем. Internet объединяет 1200 отдельных сетей, состоящих в целом из 85000 узловых компьютеров. Вирус поразил свыше 6200 компьютеров. В результате вирусной атаки большинство сетей вышло из строя на срок до пяти суток. Пользователи, чья деятельность зависит от доступа к сети, оказались полностью изолированы, то есть продуктивность их работы резко снизилась. Компьютеры, выполнявшие коммутационные функции, работавшие в качестве файл-серверов или выполнявшие другие функции обеспечения работы сети, также вышли из строя. Потеря доступа и вынужденный простой машин являются косвенными потерями, связанными с инцидентом, и оцениваются в сумму примерно 65 миллионов долларов.

Прямой ущерб состоит прежде всего в затратах времени специалистов на определение пораженных систем, дезинфицирование этих систем и восстановление их нормальной работоспособности. В первые часы атаки мало кто знал алгоритм работы вируса, поэтому большинству систем работоспособность была возвращена только после дезинфекции. Процесс дезинфекции потребовал десятки тысяч человеко-часов. После дезассемблирования вируса сотни программистов по всей стране занялись разработкой ловушек для вируса и созданием "заплат" на месте обнаруженных вирусом "дыр" в подсистеме безопасности своих систем. Эти затраты также оцениваются в десятки тысяч человеко-часов рабочего времени. Перечисленные здесь и другие виды деятельности, связанные с вирусом и продолжавшиеся иногда на протяжении недель, оцениваются в 796000 человеко-часов, что, при принятой в США средней стоимости часа работы программиста в 22 доллара, составляет свыше 17 миллионов долларов.

Время, затраченное администраторами и операторами, несколько меньше времени, затраченного программистами, и составляет свыше 300000 часов. При средней стоимости часа работы персонала этого уровня в 42 доллара, общая стоимость потерь составляет свыше 15 миллионов долларов.

Internet: 1200 сетей, охватывающих около 85200 узловых компьютеров
Инфицировано: 6200 машин (7,3% компьютеров сети).

КОСВЕННЫЕ ПОТЕРИ

	Потери машинного времени	Потеря доступа
Машины часами не имели доступа к сети	2.076.880	
Пользователи часами не имели доступа к сети		8.307.520
Накладные расходы за час	20 долл.	3 долл.
СТОИМОСТЬ	41.537.600	24.922.560

ПРЯМЫЕ ПОТЕРИ

	Время работы программистов	Время работы администраторов
Остановка, тестирование и перезагрузка 42.700 машин	64.050 час	1.000 час
Начальный анализ проблемы на 12.400 машинах	49.600 час	11.000 час
Идентификация, изоляция, удаление, чистка, восстановление работоспособности (6200 машин)	74.400 час	2.000 час
Реинфекция, удаление из сети, остановка, анализ, тестирование	62.000 час	12.000 час
Создание заплат, отладка, установка, тестирование, контроль и сопровождение	62.000 час	18.000 час
Анализ вируса, дезассемблирование, документирование (в каждой из 1200 сетей)	192.000 час	22.000 час
Исправление всех систем UNIX, тестирование, контроль	105.000 час	6.000 часов
Другие проверки, технические совещания, другие связанные с инцидентом события	187.000 час	264.000 час
ОБЩЕЕ ЧИСЛО ЧАСОВ	796.050 час	336.000 час
Средняя стоимость часа	22 долл.	42,50 долл.
ПРЯМЫЕ ЗАТРАТЫ	17.513.100	14.280.000

ОБЩИЕ ЗАТРАТЫ: 98.253.260 долл.

оставляет открытой интерпретацию доказательства преднамеренности совершенных действий;
- и на том, что закон не отражает специфику функционирования компьютеров в 1990-х годах: в законе речь идет о несанкционированных действиях в отношении отдельного компьютера, а в рассматриваемом случае была поражена многомашинная система — компьютерная сеть.

Прежде чем признать Морриса виновным, суд достаточно быстро установил несколько признаков преступления.

Самым большим мошенничеством во всей этой истории является то, что Моррис вошел в сеть для запуска вируса без авторизации.

Суд определил, что вирус Морриса причинил ущерб не менее чем на 1000 долларов, хотя власти представили более 40 свидетельств, которые в совокупности утверждают, что за счет потери производительности вирус Морриса обошелся не менее чем в 150000 долларов, а промышленная ассоциация по компьютерным вирусам оценила стоимость потерянного времени и людских ресурсов, а также сверхурочных затрат на удаление вируса из тысяч компьютеров примерно в 100,000,000 долларов.

Защита особенно настаивала на том, что вирус не поразил системы, связанные с обработкой денег; власти вынуждены были признать, что все трудности заключались в выключении машины, отсоедненнии ее от сети, повторном включении и проверке.

Далее, защита подчеркивала факт, что ресурсы в основном были затрачены на приведение иарушенной системы безопасности в состояние, предшествовавшее атаке, и дизассемблирование обнаруженного вируса с целью выяснения его опасности. При этом, поскольку электронная почта сети во время вирусной атаки не работала, многие абоненты часами пытались дизассемблировать вирус, не подозревая, что это уже сделано другими абонентами.

Другим доводом защиты был тезис о том, что доказанные разрушения не обязательно являются прямым следствием работы вируса, то есть затраты ресурсов могли быть вызваны чем-либо помимо вируса. Например, имелись показания о том, что значительное время пользователями было затрачено на проверку отсутствия в вирусе "троянских коней", "часовых бомб" или каких-либо иных разрушительных программных средств. В действительности вирус не содержал подобных элементов, поэтому защита объявила неправомочными попытки возложить на Морриса ответственность за длительные усилия, направленные на то, чтобы разрушить собственные необоснованные подозрения.

Защита подчеркнула также тот факт, что электронную почту мог использовать кто угодно для связи с кем угодно, причем без согласия на то адресата, тогда как обе эти программы в процессе работы занимают мощности и память получателя. На основании этого Гидобоии высказал утверждение, что все, кто авторизован в Internet, имеют привилегии, достаточные для использования Sendmail и Finger для связи с другими пользователями и, более того, невозможно установить, кто именно обратился к данному абоненту, поскольку для обращения к любому абоненту никакой авторизации не требуется вообще. При этом Гидобони справедливо заметил, что подразумеваемая эффективность авторизации состоит в обязательной уникальной идентификации каждого пользователя при каждом вхождении в систему, в частности в Internet.

Небольшое препирательство на суде между защитой и обвинением произошло по поводу того, имело ли бы место нарушение закона, если бы Моррис не допустил программной ошибки. Эксперт Департамента Юстиции Марк Раш заявил, что если бы вирус работал точно так, как было задумано автором, администраторы узлов сети потеряли бы время на локализацию и удаление вируса; следовательно, существенный ущерб был бы нанесен даже в том случае, если бы ошибка не была допущена.

Хотя защитник не стал оспаривать этот вывод сразу, вне зала заседаний он заявил, что если бы программа работала так, как задумывал его клиент, никакого ущерба не было бы вообще; и прояви некоторые адмистраторы немного любознательности в отношении появившегося вируса, они вполне смогли бы проанализировать его на основании достаточного количества сообщений о вирусе, поступавших от других пользователей.

Что сказал Пентагон

"...По свидетельству "Нью-Йорк Таймс", представители Пентагона, стремясь умерить опасения, что ключевые военные компьютеры уязвимы для подобных диверсий, заявили, что такой вирус не может проникнуть в секретные компьютерные сети, управляющие системами ядерного оружия и хранящими в памяти военные планы». Однако эксперты указывают, что происшедший случай свидетельствует о том, сколь уязвима компьютерная сеть".

Московская правда, 6.11.88

Вирус Морриса стал причиной очень серьезного беспокойства такого уважаемого и солидного учреждения, как Пентагон.

Не секрет, что всеобщая компьютеризация, как принято у нас говорить, американского общества не обошла стороной и эту организацию.

Компьютеры позволили создавать надежные системы связи с умопомрачительной скоростью передачи

данных, что само по себе весьма привлекательно для подобных государственных структур; компьютеры поддерживают обширные банки данных, позволяющие эффективно управлять столь масштабной организацией; компьютеры встраиваются практически во все новейшие системы вооружения, включая и ядерные, поскольку это резко повышает и эффективность применения этих систем, и оперативность их "срабатывания"; компьютеры составляют сердцевину всех навигационных систем и так называемых систем раннего обнаружения. Вспомните хотя бы пресловутую СОИ: ведь даже наши средства массовой информации все уши прожужжали о том, что эта милитаристская задумка практически во всех аспектах основывается на использовании вычислительной техники. Однако тут же наши газеты вещали, что американские военные, а с ними и "все человечество", все больше становятся заложниками компьютеров.

Как ни странно, последний вывод был не так уж далек от истины. Нарушение работы компьютерных сетей командования и управления, по мнению самих американцев, означало бы катастрофу.

Осознав всю опасность вирусов, правительство США спешно позаботилось об обеспечении надлежащей безопасности своих наиболее важных информационных систем, соединенных каналами связи с другими, незащищенными системами. По свидетельству специалистов, ответственных за защиту военных компьютерных систем, "в закрытые системы Пентагона встраиваются специальные механизмы защиты, которые работают настолько тонко, что лишь немногие люди знают, что система фиксирует каждый шаг пользователя". В частности, секретные сети передачи данных были защищены шифрующими устройствами для закрытия всех данных, поступающих на терминал и уходящих с него. Считалось, что помимо защиты от проникновения иностранных спецслужб, применение новых криптографических методов лишит компьютерных взломщиков (по крайней мере на ближайшее время) возможности совершать различные манипуляции с секретными военными компьютерными системами.

Гораздо сложнее обстоит дело с защитой публичных компьютерных сетей от неавторизованного доступа пользователей или, еще хуже, "компьютерных взломщиков" — хакеров. Особенно трудна в этом отношении защита военных сетей, соединенных с общедоступными (public domain) сетями.

Ввиду этого в 1982 году Белый Дом даже был вынужден отказаться от участия в компьютерном консорциуме 17 стран, так как выяснилось, что это позволит советским экспертам получать доступ к несекретным, но достаточно важным американским базам данных.

В соответствии с четырехлетней программой по борьбе с компьютерными вирусами DoD еще в 1987 году ужесточило режим обеспечения безопасности своих сетей. В некоторых наиболее важных компьютерных сетях были приняты следующие меры:

- максимально ограничен доступ;
- детально фиксировались все операции пользователей;
- циркулирующая в сетях информация шифровалась;
- в распоряжение пользователей сетей предоставлялись программы-вакцины, выявляющие несанкционированные изменения в программном обеспечении и данных;
- наиболее важные, содержащие секретные сведения вычислительные системы стали полностью автономными, то есть были отключены от сетей.

В январе 1981 года с целью определения пригодности предлагаемых различными разработчиками компьютерных систем для нужд DoD был создан Центр компьютерной безопасности министерства обороны США. Позднее, в сентябре 1985 года, этот центр был переименован в Национальный центр компьютерной безопасности (National Computer Security Center; NCSC) и перешел под ласковое крылышко Агентства национальной безопасности (National Security Agency; NSA).

NCSC и ныне оценивает пригодность компьютерных систем с точки зрения безопасного использования не только в вооруженных силах, но и в различных государственных учреждениях, фирмах, выполняющих государственные и военные заказы; при этом оценивание продуктов ведется по строго определенной программе, предусматривающей проверку строго определенных механизмов и средств. Это обстоятельство обеспечивает возможность сравнения рейтингов безопасности разнородных продуктов.

Оценивание осуществляется на основе стандарта, известного под названием "оранжевая книга". Согласно стандарту все компьютерные системы условно разделяются на четыре основных группы безопасности, которые в свою очередь делятся на классы безопасности. В настоящее время существуют следующие классы безопасности — в порядке увеличения гарантированности защиты: D, C1, C2, B1, B2, B3, A1.

Присвоенный компьютерной системе класс безопасности часто называют рейтингом безопасности.

Продукты, не прошедшие официальной проверки NCSC, не могут быть использованы для обработки закрытой информации в военной сфере или государственных учреждениях. Фирмы-разработчики программного обеспечения весьма заинтересованы в получении и повышении полученных официальных рейтингов безопасности для своей продукции, поскольку наличие рейтинга безопасности — необходимое условие для получения фирмой значительных государственных и военных заказов, а также хорошая реклама продукции фирмы.

Многие исследования в области компьютерной безопасности также финансировались военными.

Так что, как видите, безопасность собственных компьютерных систем весьма волновала Пентагон задолго до 1988 года.

Представьте теперь, что должны были чувствовать уважаемые американские военные, когда обнаружилось, что Milnet практически мгновенно выведена из строя вирусом!

Вирус Морриса показал уязвимость сетевых структур, поставил под сомнение надежность средств связи Пентагона и обороны США в целом. Фактически Моррис для DCA был тем же, кем и Руст для советских войск противовоздушной обороны.

Оказавшись перед фактом, что все многолетние попытки убедить налогоплательщиков в надежности военных компьютерных систем пошли насмарку, Пентагон изо всех сил постарался достойно встретить неизбежный шквал негодования. Уже 3 ноября представитель DCA, стремясь успокоить общественное мнение, публично признал факт заражения сетей министерства вирусом, однако пытался принизить значение происшедшего. Он заявил: "Вирус был выявлен в нескольких главных компьютерах, подключенных к сети Arpanet и к той части Milnet, которая содержит сведения несекретного характера... Хотя некоторые важные данные министерства обороны в той или иной мере пострадали, файлы, содержащие секретную информацию, связанную с вопросами национальной безопасности, воздействию вируса не подверглись".



Нетрудно догадаться, что число оптимистов, которые верят заявлениям официальных представителей Пентагона относительно неуязвимости секретных вычислительных систем этого ведомства, значительно убавилось. Да и ряд военных экспертов были далеки от оптимизма в оценке неуязвимости военных вычислительных систем.

Снова всплыла история с негласной проверкой степени этой "неуязвимости", предпринятой несколько лет назад группой специалистов. По образному выражению одного члена этой группы, правительственные системы "оказались похожи на швейцарский сыр, через дырки которого проникнуть внутрь можно без особого труда".

Масла в разгоравшийся огонь уничтожающей критики подлило сообщение о том, что заражения компьютерных систем Пентагона вирусом Морриса можно было избежать, если бы в свое время соответствующие органы прислушались к предупреждениям специалистов об уязвимости UNIX и рекомендациям использовать вместо нее более защищенную операционную систему VMS.

Основным и достаточно сильным доводом Пентагона был факт, что вычислительные системы и сети, в которых проходит или содержится секретная информация, являются полностью автономными, то есть они физически не соединены с внешними компьютерными системами. Тут же следовал вывод о том, что вирус Морриса не может поразить основные военные системы, поскольку он (вирус) просто не сможет в них попасть!

Как говориться, три "ха-ха": а что было бы, если бы Моррис или кто-нибудь еще запустил аналогичный вирус не во "внешней" системе, каковой может считаться Arpanet, а в любой из закрытых "автономных" систем? Чем бы тогда закончилась вся эта история?

Неважно, что вирус Морриса не уничтожал данные, а "просто" блокировал сеть. Чтобы представить себе возможные результаты блокировки ВОЕННОЙ компьютерной сети, совсем не нужно обладать чрезвычайно развитой фантазией. Бывший начальник всех информационных систем Пентагона С.Уолкер (Stephen Walker) по этому поводу заявил: "Если бы кто-нибудь

смог сделать с системой NORAD (North American Aerospace Defence Command — Командование аэрокосмической обороны Северной Америки) то же самое, что сделал Моррис с Arpanet, ущерб был бы огромен". Подумайте сами, что значит, например, вывод из строя военной системы предупреждения о нападении даже на несколько минут с учетом, что так называемое "время подлета" к американскому континенту межконтинентальных ядерных ракет составляет около получаса. А ведь вирус Морриса заблокировал Arpanet минимум на сутки!

Несмотря на очевидный промах с вирусом Морриса, военные специалисты оценивают проблему вирусов весьма дальновидно. Например, один из них заявил: "Возможные негативные последствия наступательного использования вирусов настолько велики, что по разрушительной силе и площади поражения я сравнил бы их с ядерным или химическим оружием".

Что же предпринял Пентагон после столь дорого стоившего ему инцидента?

После упоминавшейся ранее встречи, состоявшейся 8 ноября, представитель Пентагона заверил, что "процесс совершенствования программ идет полным ходом". Было объявлено также о дополнительных мерах, которые будут приняты для обеспечения безопасности военных компьютерных сетей:

- еще более ужесточается процедура доступа пользователей к аппаратным и программным средствам;
- внедряется новое сложное сетевое программное обеспечение, которое "выявляет и блокирует" все попытки вируса проникнуть в систему;
- вводится более жесткий стандарт безопасности для военных компьютеров.

Возможные негативные последствия наступательного использования вирусов настолько велики, что по разрушительной силе и площади поражения их можно сравнить с ядерным или химическим оружием.

В начале 1989 года стало известно, что DARPA планирует начать разработку новой национальной компьютерной сети Defence Research Internet (Сеть исследований Министерства обороны — DRI), которая придет на смену сети Arpanet. Сеть DRI создаст информационно-справочную базу для проводимых Управлением исследований и разработок в области систем связи, контроля и управления, обеспечит доступ специалистов управления и корпораций-подрядчиков к новому поколению систем параллельной обработки данных. Кроме того, сеть будет своего рода полигоном для отработки новых подходов и концепций развития сетевых вычислительных структур DoD. В процессе разработки новой сети будет уделено большое внимание созданию эффективных средств обеспечения безопасности сети и циркулирующих в ней данных, способных гарантировать надежную защиту от вирусных атак и других нарушений режима секретности.

В настоящее время разрабатываются так называемые "доверительные системы" (trusted systems), которые надежно гарантируют, что пользователь получает доступ только к такой информации, какую ему положено читать, и может совершать с системой такие операции, на какие он имеет официальное разрешение. Специалисты утверждают, что "доверительные системы" позволят более грамотно объединять компьютеры в информационные сети без снижения уровня безопасности. Отсутствие таких систем — главное препятствие к более эффективному использованию компьютеров в закрытых информационных сетях. До окончания их разработки информационные сети нельзя считать полностью безопасными с точки зрения возможности несанкционированного проникновения.

Таким образом, риск утраты секретной информации пока по-прежнему реален.

А теперь относительно сделанного мною несколько раньше намека на крайне щекотливое положение, в котором оказался NCSC.

Как я уже объяснил, NCSC обязан рождением Пентагону, так что "слава" одного из них косвенно "осеняет" и другого. Но на деле все обстоит гораздо интереснее: Роберт Таппан Моррис является сыном одного из наиболее известных правительственных экспертов по компьютерной безопасности — научного руководителя NCSC! Каково!

Эффект будет более поразительным, если учесть, что во время доклада Конгрессу в 1983 году именно Моррис-старший говорил о возможности обмана технически образованными новичками специалистов по безопасности как о "совершенном нонсенсе".

Но и это еще не все! Оказывается, баловство с компьютерными вирусами для Моррисов чуть ли не семейная традиция: Моррис-старший, работая в 60-х годах в Bell Laboratories, принимал участие в создании игры, основанной на компьютерном вирусе. Ядром этой игры, которая называлась Core Wars, была программа, которая могла размножаться и пыталась разрушить программы других игроков.

Моррис-отец прокомментировал события следующим образом: "Некоторые считают, что суть проблемы состоит в том, что в США выросло новое поколение хорошо разбирающихся в технике людей, которые при желании извлечь из компьютерной системы секретные данные в состоянии преодолеть все барьеры, создаваемые специалистами по защите информационных систем крупнейших американских корпораций и министерства обороны... Однако все не так просто".

По мнению Морриса-старшего, именно новый вирус позволил выявить эти уязвимые места, поэтому его сын якобы сыграл "конструктивную роль" и содействовал совершенствованию методов организации компьютерных сетей и управления ими.

Тем не менее инцидент существенно не отразился на карьере Морриса-отца. По крайней мере в начале 1989 года он был избран в специальный консультативный совет при Национальном институте стандартов и технологии (NIST; бывшее Национальное бюро стандартов) и курирующем его министерстве торговли. Этот совет образуется ежегодно в соответствии с положениями Закона о безопасности компьютеров (Computer Security Act of 1987). Задачами этого совета являются выработка заключений и рекомендаций по вопросам безопасности вычислительных систем правительственных ведомств США, а также решение проблем, возникающих в процессе разработки и внедрения новых стандартов защиты несекретной, но важной информации, хранящейся или циркулирующей в этих системах и тому подобного.

Что об этом думали

"Значение происшедшего не в конкретных сиюминутных последствиях. Инцидент этот — напоминание о "Челленджере" и Чернобыле. Мы создали такие сложные системы, что порой не можем полностью удержать их под своим контролем".

Дж.Уайзенбаум, сотрудник MIT

"Молодому Моррису, на удивление, симпатизируют очень многие — это явно прослеживается по потоку сообщений. Думается, настала пора исключить из компьютерной профессии позицию «ребенок есть ребенок». Или, как лучше сказано в Библии: «Когда я был ребенком, я говорил как ребенок, я думал как ребенок, я рассуждал как ребенок; когда я стал взрослым, я отказался от детских путей»".

Филипп Гарднер, полковник в отставке

"Мы не в состоянии учиться на собственных успехах, но зато отлично учимся на собственных промахах".

Генри Петровски

Пятница, 11 ноября 1988 года.

09:27 RISKS Обсуждение непригодности UNIX для использования в системах с более-менее серьезными требованиями к безопасности (медицинские, коммерческие и государственные).

19:55 VIR Призыв компьютерных профессионалов наказать злоумышленника за неэтичное поведение. Импульсивный сомнительный комментарий Тэда Коппеля из NightLine, что Роберт Моррис имеет будущее в области компьютерной безопасности.

Научная и компьютерная общественность разделилась в оценке вины Морриса.

Администраторы систем, проведшие дни в борьбе с вирусом, готовы были линчевать Морриса, попадись он в их руки. Немудрено: во-первых, обнаружение неизвестного вируса вызвало у многих нешуточный испуг за свои системы, а во-вторых, атака стала причиной внеплановой работы многих ученых и исследователей, а также лишила еще более широкий круг пользователей возможности работать с сетями, объединяющими научные центры страны. Естественно это замедлило ход научных и инженерных исследований и разработок.

Результатом вирусной атаки можно считать также отмеченное многими очевидное сокращение обмена информацией посредством компьютерных сетей между университетами, лабораториями и так далее.

Тем не менее многие эксперты по компьютерной безопасности заявили, что вирус стал важной демонстрацией потенциальной уязвимости компьютерных систем и последствия этой атаки являются, безусловно, положительными в том смысле, что атака вызвала всеобщую тревогу пользователей и их заинтересованность в разработке и внедрении средств защиты от возможных атак такого рода в будущем.

На это оппоненты ответили доводом, что усиление средств компьютерной безопасности повредит экономике страны.

Существует мнение, что данный вирус в действительности является представителем полезного вида сетевого программного обеспечения. Этот класс программ может быть с успехом использован для обеспечения связи между системами и синхронизации их совместной работы. Эти же программы могут послужить и для выявления некорректной работы сети, выполнения задач с большим объемом вычислений, распараллеленных по нескольким системам так, как если бы все эти системы представляли собой единое целое, а также в качестве скоростной электронной почты.

Инцидент подтвердил высказанную ранее мысль, что, если ЭВМ связана с "внешним миром", то есть подключена к сети, то не может быть и речи о гарантированной защите информации.

Большинство причисляет Морриса к многочисленному племени хакеров. Этот термин первоначально использовался в отношении компьютерной субкультуры, однако в настоящее время употребляется в основном как синоним компьютерных взломщиков. В более широком, и думается, более правильном понимании хакер — это человек, достигший значительных вершин в области компьютерных наук и фанатично преданный

своему компьютерному призванию. Хакеры получают настоящее наслаждение от удачно написанной программы (не важно, своей или чужой); коллекционируют всевозможного рода ухищрения и неожиданные решения в программировании. Именно хакеры обычно первыми осваивают до тонкостей новые программные пакеты и системы, равно как именно хакеры стремятся выжать из новой техники все, на что она способна.



Если понимать термин "хакер" в предложенном смысле, то я согласен считать Морриса хакером, поскольку он не стремился напакостить, а хотел разобраться с обнаруженными им возможностями. Кстати, профессор Корнеллского университета — специалист в области компьютерной техники — признал, что Моррис вполне усвоил учебную программу и поэтому "может считаться хакером с Гарвардским образованием".

Что же касается электронных взломщиков, то они в последнее время серьезно заинтересовались компьютерными вирусами как одним из наиболее эффективных и безликих способов внедрения в системы. В ряде публикаций даже появились рекомендации и инструкции, подробно объясняющие, как составить программу-вирус.

Как вы уже знаете, АНБ и ФБР пытаются ограничить распространение небезопасной информации. Однако несмотря на все усилия АНБ и ФБР, копии вируса Морриса стали для хакеров эталонами искусства программирования. Они учатся на ней, пытаются ее усовершенствовать. Предполагается, что остатки вируса Морриса по-прежнему существуют в отдельных узлах сети Internet. Кроме того, считается, что около 1000 человек в США владеют исходным кодом вируса Морриса. Так что, если кто-нибудь из обладателей исходного кода держит его не только для коллекции, мы скоро сможем снова наблюдать нашествие этого вируса, но на этот раз он будет работать лучше, в том смысле, что принесет гораздо больше вреда.

В развитие мысли о благотворном влиянии хакерства можно привести существующее среди специалистов мнение, что США удерживают мировое лидерство в вычислительной технике и программном обеспечении благодаря таким людям, как Моррис. Пытаются даже обосновать такую зависимость: все успехи США в развитии вычислительной техники связаны с одиночками-индивидуалистами.

В качестве доказательства приводится факт, что удачные программы, получившие широкое распространение, — Wordstar, Lotus 1-2-3, Visicalc — разработали одиночки или небольшие группы. Полезно вспомнить также, что два друга — бывшие "взломщикн" Возняк и Джобс изобрели персональный компьютер, изменивший лицо Америки и всего мира.

Зато программы, созданные большими коллективами программистов (Visi-On, Jazz), оказались неудачными и привели к банкротству фирм-разработчиков.

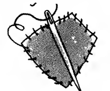
До недавнего времени социологи рассматривали хакеров — в основном, конечно, из ультралевого крыла хакерства — как своего рода неудачников, пытались им помочь, трудоустроить в области информатики, являющейся предметом их интереса. Однако из этой затеи ничего не вышло. Нелегальное проникновение стало для многих представителей субкультуры своего рода наркотиком. У некоторых из них удивительные судьбы.

Так, талантливый программист Дж.Дрепер в 70-е годы провел шесть месяцев в тюрьме за "нелегальное использование телефонной сети". После выхода из заключения он написал программу Easy Writer — весьма популярную в США программу обработки текстов, одну из первых, которая была использована в персональных компьютерах корпорации IBM. Несмотря на финансовый успех программы, он не так давно был снова осужден за совершение очередного "компьютерного преступления".

Что же касается вопроса о вреде, наносимом хакерством, то думается, гораздо большую опасность для компьютерного мира представляют собой миллионы пользователей-непрофессионалов, получивших в свои руки мощную микротехнику. Именно некомпетентностью пользователей во многом объясняется столь широкое распространение компьютерных вирусов. Ведь вирусы создавались и раньше, но раньше не возникало "компьютерных эпидемий".

Например, Чейсик из лаборатории JPL, который теперь тоже борется с компьютерными вирусами, еще в 70-х годах писал вирусные программы в колледже. По его словам, это было своеобразным тестированием на зрелость, которое должен был пройти каждый программист. В этом не было никакого злого умысла. Такие программы могли мешать компьютерным пользователям в реализации их замыслов, но ничего на разрушали.

Когда Моррис-старший и Чейсик создавали свои вирусные программы, компьютерный мир был сплоченным и высокодисциплинированным сообществом. В современном мире персональных компьютеров большинство пользователей не являются компьютерными профессионалами. Это гораздо более хаотичное образование, являющееся привлекательной мишенью для создателей вирусов.



Еще одна интересная теория относительно современной (и самой совершенной) технологии развития аппаратных и программных средств: хакеры отыскивают уязвимые места систем и сетей — затем специалисты заделывают эти дыры. Примером такой "технологии" опять же является дело Морриса.

В силу всего сказанного не покажется крамольной мысль, высказанная Джозефом Хайландом, что ком-

пьютерные хакеры являются достоянием нации и национальной головной болью.

Как ожидается, одним из наиболее важных последствий инцидента с вирусом Морриса будет резкое изменение отношения к проблеме компьютерной безопасности. Ряд экспертов, например, из консультативной фирмы Ernst & Whinney, настаивали на пересмотре всего используемого в стране программного обеспечения. Другие отвергают эту идею, полагая, что она обойдется слишком дорого, займет много времени и ограничит право законных пользователей на совершенствование, исправление и адаптацию программного обеспечения, негативно повлияет на формирование информационного общества.

Наиболее осторожные эксперты считают, что решение этой проблемы должно осуществляться на основе анализа соотношения "стоимость/эффективность", поскольку существует определенный предел, за которым дальнейшее повышение уровня безопасности оказывается не только неэкономичным, но и неэффективным. По образному выражению одного из них, "мы можем уподобиться жильцу, который, постоянно совершенствуя и усложняя систему защиты своего жилища, в итоге не сможет в него попасть".



Эпилог

Итак, почему же я предлагаю запомнить 2 ноября 1988 года.

Прежде всего потому, что начавшийся в этот день инцидент связан с компьютерным вирусом, считающимся в настоящее время наиболее опасным.

Следующим, весьма спорным, моим утверждением будет то, что вирус Морриса стал концом целого периода в существовании такого явления, как компьютерные вирусы. Компьютерные вирусы становятся все изощреннее и защищаться от них — как и бороться с ними — становится все сложнее. С другой стороны, появляются все более мощные средства защиты от компьютерных вирусов. Все это позволяет предполагать, что в будущем вирусы будут создаваться хорошо подготовленными специалистами, и преследоваться при этом будут вполне определенные цели. Как сказал Стивен Росс, старший администратор фирмы бухгалтерских услуг и консультаций Delloitte, Haskins & Sells: "Эпоха вирусов-шутих, не имеющих каких-либо определенных задач или целей, кончается, перед нами — эпоха ориентированных вирусов".

Инцидент с вирусом Морриса затронул также столь многие проблемы, связанные с компьютерами, что на примере этого дела стоит внимательно и долго учиться, чтобы избежать повторения выявленных инцидентом недостатков в родных пределах.

Что касается самого Морриса-младшего, то окончательно он был приговорен к трем месяцам тюрьмы и денежному штрафу в 270 тысяч долларов, а вдобавок был исключен из Корнеллского университета.

Думается, однако, не стоит слишком сильно печалиться о его дальнейшей судьбе: уже в январе 1989 года президент Alliant Computer Systems Corp. Рон Грюнер (Ron Gruner) в ответ на вопрос журнала Computerworld по поводу намерений компании нанять на ра-

боту Р.Т.Морриса, сказал следующее: "Некоторые очень известные в (компьютерной) промышленности люди начинали с хакерских проделок в самых изощренных формах".

И.Моисеенков

По материалам:

E.Spafford "The Internet Worm: Crisis and altermach", Communication of the ACM, vol.32, June 1989.
J.Rochlis, M.Eichin "The Internet Worm: With Microscope and Tweerers: The worm from MIT's Perspective", Communication of the ACM, vol.32, June 1989.
C.White "Viruses and Worms: Campus Attacks", Computer & Security, №8, 1989.
P.Gard "Internet Worm", Computer & Security, №8, 1989.
E.Spafford "The Internet Worm Programm: An Analysis", ACM Commitee Report, 1989.
H.Hidhland "Random bits & bytes", Computer & Security, №8, 1989.
I.Shuman "Case about not catched virus", UNIX Today, Febriary 5, 1989.
M.Alexander "Dissecting the Anatomy of Worm", ComputerWorld, November 14, 1988.
P.Fites, P.Johnston, M.Kratz "The Computer Virus Crisis", 1989.
D.Stiff, P.Carroll "One mistake and "harmless" miscief brought notoriety to Robert Morris Jn.", The Wall Street Journal, November 1988.
Financial Times, November 5-6, 1988, p.1-2.
Financial Times, November 10, 1988, p.10.
Nature, №6195, 1988, p.97.
Nature, №6197, 1988, p.301.
Times, November 5, p.4.
Times, November 7, p.10.
Times, November 8, p.10.
Times, November 9, p.4.
Japan Times, November 6, 1988, p.3.
Japan Times, November 7, 1988, p.1, 7.
Sunday Times, November 6, 1988, p.19.
New York Times, November 26, 1988, p.1, 28.
Лазарев А. Эти доверчивые компьютеры./ Эхо планеты. 1988, ноябрь.

Фирма SHARP начала поставки в СССР своего лазерного принтера JX-9500E стоимостью 1340 долларов, ориентированного на использование в системах настольных издательств. JX-9500E — один из самых компактных в мире лазерных принтеров — примерно на треть меньше HP-IIP и почти в два с половиной раза компактнее HP-III.

Принтер может эмулировать работу HP Laser Jet Series II, Epson FX-80, IBM Graphics Printer, IBM Proprinter и Diablo-630. Выбор эмулятора осуществляется из меню на панели управления принтером.

Скорость печати составляет 6 страниц в минуту, максимальный формат бумаги A4, разрешающая способность этого принтера 300x300 точек на дюйм. Обращает внимание то, что первая страница печатается после очень небольшой задержки. Оперативная память принтера — 512 Кбайт с возможностью расширения до 4.5 Мбайт. Контроллер построен на процессоре 68000.

Принтер использует шесть резидентных фонтов. Возможно расширить их набор с помощью дополнительных картриджей.

Кроме того, фирма SHARP предполагает поставлять на советский рынок цветной струйный принтер JX-730 и цветной сканер JX-450.

Принтер использует 48-струйную 4-цветную печатающую головку и позволяет воспроизвести 7 цветов. Разрешающая способность составляет 180x180 точек на дюйм. Максимальная ширина изображения 345 мм. Принтер может найти применение в архитектурных бюро или мастерских дизайнеров.

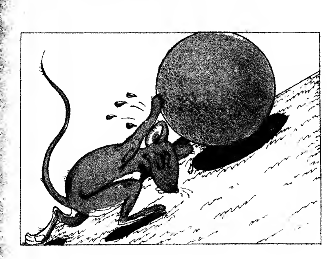
Сканер обеспечивает работу с разрешением до 300x300 точек на дюйм и воспроизводит свыше 260 тысяч оттенков. Кроме того, возможно использование сканера в черно-белом режиме с 256 градациями серого или в чисто "штриховом" режиме.

Сканирование изображений возможно как с отражающих, так и с прозрачных оригиналов. В первом случае максимальный размер составляет 431x297 мм (формат A3), во втором — 297x210 мм (A4). Возможна подстройка цветовых характеристик сканера для работы в паре с конкретным монитором или принтером.

Цветоделение осуществляется посредством трех люминесцентных ламп с соответствующим спектром излучения и специальной схемы с зарядовой связью фирмы SHARP.

Как сообщил нам представитель фирмы Digital Research, фирма готова выпустить новую версию операционной системы DR-DOS. Это событие может произойти уже в конце августа этого года.

Существующая сейчас версия DR-DOS 5.0 по многим параметрам превосходит недавно выпущенную фирмой Microsoft операционную систему MS-DOS 5.0, а новая версия будет обладать еще более высокими характеристиками.



На страницах "КомпьютерПресс" часто эксплуатируется тема общения — общения между людьми. В этой статье мы несколько изменили ее направленность. Давайте поговорим с вами сегодня об общении с компьютерами.

Устройства ввода информации

Введение

Как часто мы судим о человеке по его словам, а не по его делам. Для нас важнее то, что мы говорим своему собеседнику и что он нам отвечает. И лишь редкие люди, в основном анатомы и психологи, захотят разбираться в том, как устроен человек, его тело и душа. Сказанное относится и к компьютерам. Думаю, не ошибусь, если скажу, что девять десятых пользователей персоналок уделяют значительно больше внимания не техническим подробностям устройства аппаратного обеспечения, а тому, насколько удобно им работать с техникой, выполняя свои сугубо прикладные задачи. Что называется — каждому свое. Вот тут-то и всплывает проблема, решение которой зачастую может повысить производительность работы с компьютером значительно больше, чем, скажем, применение нового микропроцессора. Речь идет о совершенных устройствах ввода информации (УВИ). Ведь их функциональные и технические возможности и ограничения — это, по существу, функциональные и технические возможности и ограничения всей системы в целом. И от того, насколько удобны эти устройства, как они выглядят, нравятся ли они нам, зависит и настроение пользователя, и в конечном итоге его производительность труда (последнее, по мнению классиков, самое главное, самое важное...).

Компьютер информирует пользователя и (или) других потребителей (это могут быть, например, другие вычислительные системы) о результатах своей работы в форме цифровых кодов, принимающих форму графических или текстовых изображений на экране дисплея, звуковых сигналов или даже механических воздействий, но ничего в принципе не происходит до тех пор, пока пользователь не обратится к тому или иному УВИ и, таким образом, не заставит компьютер "услышать" себя.

Для построения УВИ используются различные принципы, но характерной их чертой является определенная специализация: устройство конкретного типа лучше всего приспособлено к вводу в машину информации определенного характера.

К наиболее распространенным компьютерным УВИ относятся клавиатуры, манипуляторы для управления курсором — типа мыши и трекбола, сканеры и системы оптического распознавания символов (OPC). Реже встречаются различного рода средства ввода графической информации (графические планшеты, диджитайзеры), световые перья и сенсорные экраны, которые относятся к средствам специализированного назначения.

Рынок УВИ весьма разветвлен, и выпуском каждого типа устройств занимается большое число фирм. Однако устройства одного и того же типа, выпускаемые разными фирмами, различаются весьма незначительно. Клавиатуры относятся к устройствам универсального применения и чаще всего включаются в состав поставки компьютерной системы как ее неотъемлемая часть — при этом цена клавиатуры обычно включается в цену системы. Мыши, которые уже приобрели на

рынке статус универсальных устройств (в дополнение к традиционной нише применений, каковой являются системы, работающие с графикой), также довольно часто предлагаются фирмами—поставщиками компьютерных систем — иногда в комплекте поставки компьютера, но чаще за отдельную плату — как дополнительное устройство.

Сканеры для считывания изображения, устройства OPC, графические планшеты и другие специализированные УВИ обычно не рассматриваются в качестве главных средств ввода информации в компьютер. Чаще всего это важный или просто полезный инструмент для выполнения специфических задач, прежде всего в издательских системах и системах автоматизированного проектирования. Поэтому главными действующими лицами на рынке УВИ этих типов являются не производители компьютеров, а т.н. третьи фирмы (под первой и второй понимаются покупатель и продавец компьютера). Исключение составляют лишь несколько крупнейших фирм—поставщиков компьютерных систем, включая, прежде всего, IBM и Hewlett-Packard.

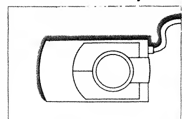
Клавиатуры

Кто не знает, что такое клавиатура, поднимите руки. Я думаю, на лес поднятых рук можно надеяться с весьма сомнительной долей вероятности, разве что где-нибудь на общем собрании племени Мумба-Юмба. И хотя про клавиатуры вместо меня с тем же успехом вам расскажет все необходимое каждый первоклассник, все же, для полноты изложения, кратко остановимся и на них. Ведь клавиатуры остаются главным средством ввода практически во всех задачах прикладного характера — электронных таблицах и базах данных. Это ведущее положение они заняли еще со времен первого появления компьютеров. Нет, вернее, значительно раньше. Вспомните, ундервуды и ремингтоны. Но прогресс не стоит на месте, и вот на смену пишущим машинкам приходят мощные текстовые процессоры, где клавиатура все также незаменима. А дальше — больше. Захотелось на компьютере не только набрать текст, но и сверстать его, вставить туда картинки, которые, правда, либо рисуются мышкой или трекболом, либо сканируются. Но и здесь в той или иной степени остается клавиатурный ввод. Так что, как ни крути, а без клавиатуры не обойтись.

В техническом аспекте компьютерная клавиатура представляет собой совокупность простых 2-позиционных переключателей — либо механических кнопок, либо электронных бесконтактных, т.н. "мягких" переключателей. При нажатии на клавишу происходит замыкание электрической цепи и в компьютер посылается дискретный электрический сигнал. При этом срабатывание переключателя каждой клавиши или определенных сочетаний переключателей нескольких одновременно нажимаемых клавиш вызывает посылку в машину уникальных цифровых кодов. Существует несколько стандартизованных систем кодирования. Наиболее популярной в области персональных компьютеров (ПК) сейчас является система ASCII

Трекбол PC-Trac

Несмотря на сравнительно большие размеры, этот трекбол фирмы Micro Speed удобен в работе. Форма его корпуса такова, что кисть и запястье во время работы находятся в том же положении, что и при манипулировании мышью. Устройство располагает тремя кнопками, из которых две внешние охватывают шар и поэтому очень удобно нажимать быстрыми прикосновениями, а на средней есть выступ (букля), благодаря которому она (кнопка) легко нащупывается пальцем.



Pc-Trac можно без всяких хлопот включить в систему, располагающую программным обеспечением для мыши. В комплект поставки входит дискета с программой-утилитой Keymap, резидентной программой для "привязки" двадцати широко распространенных прикладных программ, которые в оригинальных версиях работу устройств управления курсором не поддерживают.

Все опрошенные пользователи трекбола Pc-Trac указали в качестве основной причины своего перехода от мыши к трекболу то, что последний требует меньше места на столе, где стоит компьютер. Кроме того, пользователи говорят, что трекбол удобнее мыши. Им, в частности, легче рисовать на экране. Есть опыт использования описываемого устройства с графическими пакетами —системами типа САПР и САИ (системы автоматизированного проектирования и системы автоматизированного инжиниринга), а также с широко известной системой Word Perfekt.

Однако для целей САПР, для работы в системах Lotus 1-2-3, Quatro Pro и им подобных лучше подходит близкий "родственник" трекбола Pc-Trac — устройство Fast-Trap, у которого в верхней части корпуса кроме кнопок имеется колесико, дающее возможность маневрировать курсором в третьем измерении. Речь может, например, идти о перемещении по слоям микросхемы при работе в САПР, предназначенной для разработки ИС.

Разрешающая способность устройства 200 cpi. Цены: модель с последовательным интерфейсом — 200 долл.; модель для компьютеров типа PS/2 — 119 долл.

(Американский стандартиый код для обмена информацией).

Обычио компьютерная клавиатура содержит порядка 60 клавиш с буквами, цифрами, знаками пунктуации и другими символами, встречающимися в печатных текстах. В этой части клавиатуры отражается языковая принадлежность пользователя. Кроме того, клавиатуры имеют еще 30-40 клавиш, предназначенных для управления компьютером и исполненнем программ. Эта часть клавиатуры от языковой принадлежности пользователя не зависит. Сюда входят клавиши управления курсором, функциональные клавиши, клавиши ускореиного перемещения по экрану, клавиши для выдачи ряда специальных команд, воспринимаемых аппаратными и программными средствами компьютера.

Клавиатуры разиых фирм—изготовителей различаются некоторыми сравнительно маловажными характеристиками, например, величиной усилия нажатия, наличием и величииой тактильной или звуковой обратной связи (под обратной связью в данном случае понимается уведомление пользователя о срабатывании переключателя — это может быть воспринимаемый пальцем щелчок или звук, который появляется при полном нажатии), наличием встроенных в клавиши световых индикаторов для отображения текущего состояния переключателей. Кроме того, имеются различия в габаритах клавиатур, размерах и расположении отдельных клавиш.

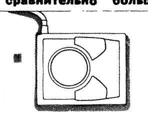
Стандартная компьютерная клавиатура представляет собой плоский конструктив толщнной примерно 50, длиной 460 и шириной 200 мм. При конструировании клавиатур, в зависимости от типа и (или) главного направления применения компьютера, реализуют различные варианты размещения клавиш. Например, портативные компьютеры (лэптопы и ноутбуки) оснащаются более легкими, компактными клавиатурамн. Имеются и клавиатуры с сокращенным числом клавиш — они применяются в магазинах, на складах, в портах и т.п., где критически важными являются размеры и вес устройства.

В большинстве случаев клавиатуры являются взаимозаменяемыми: клавиатуру от одного ПК можно подключить к другому. Исключение составляют клавиатуры для ПК семейства Macintosh, которые отвечают требованиям уникального стандарта фирмы-изготовителя.

Если говорить о ПК фирмы IBM, то системы кодирования, назначения и размещения клавиш на клавиатурах компьютеров класса XT и т.н. улучшенных клавиатурах машин классов AT и PS/2 несколько различаются. Стандартная XT-клавиатура содержит 84 клавиши, из которых 10 функциональных и 17 клавиш цифровой клавиатуры. Улучшенная AT-клавиатура содержит 101 или 102 клавиши, в том числе 12 функциональных. Клавиши для ввода цифр и управления курсором конструктивно разнесены.

Трекбол Expert Mouse

Этот трекбол фирмы Kensington отличает малая площадь осноаания, и а то же время он имеет шар сравнительно большого диаметра. Двумя кнопками устройстаа, которые расположены вблизи от "ватерлинии" шара, легко пользоваться большим и безымянным пальцами (или мизинцем); при этом оамые длинные пальцы — указательный и средний — остаются свободными для манипулирования шаром.



К настоящему времени имеется только версия для компьютеров типа PS/2, но, безусловно, должна появиться и аерсия для машин типа PS. Программное обеспечение, которое поставляется с устройстаом, дает аозможность адаптироаать его к требованиям конкретного заказчика: менять местами кнопки (по их назначению) — это бывает необходимо, например, когда устройством должен будет пользоваться леаша; эмулировать работу 1-, 2- и 3-кнопочной мыши; определять режим работы одной из кнопок (возможны два режима: кратковременного нажатия и с фиксацией — т.е. западающей кнопки).

Все эти изменения вы можете внести ао аремя установки (инсталяции) программы — либо по подсказкам, выдааааемым в форме меню, либо путем ваода командных строк заданного формата. Программа обеспечиаает также аозможность пользоваться трекболом при работе с рядом распространенных программ, при разработке которых применение устройств управления курсором не предусматривалось.

Те, кто имеет опыт эксплуатации этого трекбола (а они работали а среде таких программ, как Norton commander, Window/286, Window/386), отмечают удобные очертания его корпуса и удачное расположение кнопок.

Разрешающая способность устройства 200 cpi. Цена модели для компьютеров типа PS/2 — 169.95 долл.

Мыши и трекболы

Мышь представляет собой еще одно широко распространенное УВИ, облегчающее пользователю работу со многими прикладными программными системами и делающее ее более простой и эффективной. В основной своей функции мышь является устройством управления положением курсора на экране монитора: перемещение мыши по гладкой поверхности (или по поверхности специального планшета) автоматически

преобразуется в пропорциональное по величине и совпадающее по направлению перемещение курсора по экрану. Встроенные в тело мыши клавиши позволяют пользователю подавать в ПК сигналы о том, что курсор достиг требуемого положения, и тем самым выбирать те или иные объекты (например, пункты меню), перемещать их по экрану, вызывать одни объекты и убирать с экрана другие, а также эмулировать действие управляющих клавиш клавиатуры.

Своей популярностью мышь обязана главным образом растущему спросу на прикладные графические программные системы, а также распространению графического интерфейса пользователя, когда широко применяются мнемонические изображения объектов — пиктограммы. Возможности клавиатуры явно не согласуются с характером работы пользователя в такой "изоориентированной" среде. Поэтому и возникла потребность в другом средстве связи пользователя с компьютером. Самым популярным из различных модификаций этого средства оказалась мышь, которая делает очень удобным манипулирование такими широко распространенными в графических пакетах объектами, как окна, меню, кнопки, пиктограммы.

При конструировании мышей применяются механический, оптический или оптомеханический принципы действия.

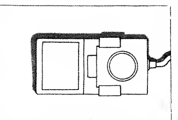
В корпусе механической мыши имеется шар сравнительно большого диаметра, который вращается, когда пользователь перемещает тело мыши по поверхности стола. Шар приводит во вращение два ролика (ось вращения одного из них горизонтальна, второго — вертикальна). Те в свою очередь приводят в действие два механических дешифратора, которые посылают свои выходные сигналы схеме интерфейса с мышью, имеющейся в компьютере. Последняя обеспечивает перемещение курсора по экрану монитора.

Оптомеханическая мышь отличается от механической только тем, что вместо механических дешифраторов используются оптические, и сигналы посылаются в компьютер в результате срабатывания не механических, а бесконтактных оптических переключателей (т.е. срабатывающих при падении на них светового потока).

Оптическая мышь вообще не имеет движущихся частей. Перемещение воспринимается оптическими датчиками (встроенными в корпус устройства) в процессе их смещения относительно поверхности специального планшета. Механическая и оптомеханическая мыши не требуют специального планшета — их можно перемещать по поверхности стола, по бумаге, стене и т.п. Однако они менее защищены от попадания пыли и грязи по сравнению с оптическими устройствами. В общем случае оптическая мышь более долговечна, но требует свободного места для размещения планшета.

Трекбол Mouse-trak

Фирме ITAC придется еще немало потрудиться, чтобы оправдать рекламу своего трекбола Mouse-trak как "самого лучшего трекбола в мире". Во-первых, для его инсталяции требуется достаточно утомительная процедура. Для того чтобы поставить эмулирующую программу Microsoft Mouse (либо аналогичную ей программу Mouse Systems) или задать клик-кнопку необходимо выставить в заданные положения несколько переключателей и перезапустить процессор кнопкой "reset". Боже вас упаси от попытки поменять местами назначение кнопок устройства — для этого его надо полностью разобрать.



Удачной находкой в этом устройстве является специальная площадка с подушечкой, на которой может при работе покоиться кисть руки. В то же время от манипулирования тремя маленькими неудачно расположенными кнопками трекбола очень быстро устают пальцы. Хотя курсор перемещается по экрану довольно плавно, но все же рисовать трекболом трудно — даже при переключении его на самую малую скорость.

Трекбол Mouse-trak поставляется с резидентной программой, которая выдает на экрвн меню, позволяющие пользователю рвботать в среде ДОС, в системах Lotus 1-2-3 и Word Perfect 4.2.

Описываемый трекбол нашел применение на ряде рыболовецких судов — в составе поставляемой под ключ системы (на 386-ом процессоре) с пакетом программного обеспечения Sca Plot (на базе системы Windows), который предназначен для формирования электронных навигационных карт. В характерных для данного применения жестких условиях трекбол проявил себя очень хорошо. Использовать мышь на судах практически невозможно, во-первых, из-за качки, а во-вторых, из-за грязи, которой твм всегда больше, чем достаточно.

Разрешающая способность устройства 200 cpi. Цены: 2-кнопочная модель с последовательным интерфейсом и модель для компьютеров типа PS/2 — 169 долл.; аналогичная 3-кнопочная модель — 179 долл.; 2-кнопочная модель с параллельным интерфейсом — 189 долл.; то же, но 3-кнопочная модель — 195 долл.

Известны модели мышей, в которых есть возможность менять соотношение скоростей перемещения мыши и курсора — это т.н. мыши динамического действия. В некоторых случаях реализовано такое реше-

ние: первые 1-2 дюйма перемещения мыши (1 дюйм = 25,4 мм) вызывают медленное, "тонкое" смещение курсора, а дальнейшее перемещение приводит ко все более непропорциональному ускорению движения последнего. Есть модели с постоянным, но задаваемым извне соотношением перемещений, т.е. есть возможность устанавливать величину этого параметра при настройке программного пакета, с которым предполагается работать в данный период времени.

Мыши обычно оснащаются одной, двумя или тремя кнопками, на которые пользователь может нажимать либо кратковременно, либо длительно. При этом в компьютер посылаются сигналы, уведомляющие его о том или ином решении пользователя. Одна из кнопок (она обязательна) служит для фиксации выбора пользователем того или иного объекта на экране монитора. Вторая и третья кнопки (если они поддерживаются аппаратными и (или) программными средствами компьютера) могут использоваться для эмуляции клавиш клавиатуры или выполнения других функций. Однако стандартов на аппаратно-программное обеспечение мышей до сих пор нет. Это означает, с одной стороны, что ряд прикладных программных систем ориентирован только на 1-кнопочные мыши, а с другой, что возникают серьезные трудности при попытке "привязать" мыши некоторых моделей к тем или иным прикладным программным системам.

Что касается интерфейса с мышью, то он реализуется в двух вариантах: 1) сама мышь поставляется как неотъемлемая часть компьютера — в этом случае никаких дополнительных средств к компьютеру не требуется (такого решения придерживается фирма Apple в своем семействе ПК Macintosh; 2) мышь поставляется как отдельное средство. По последнему пути идет фирма IBM. В этом случае интерфейс может принимать форму стандартного последовательного порта; специальной платы-адаптера, устанавливаемой в гнездо расширения ввода-вывода; специальной платы-адаптера, которая устанавливается в полуразъем-соединитель для клавиатуры и имеет полуразъем для подключения последней. Для пользователя, с точки зрения функциональных возможностей и качества работы мыши, все эти варианты равноценны.

Хотя никаких официальных стандартов на мыши до сих пор не разработано, де-факто на рынке существуют три стандарта, которым фирмы — изготовители мышей всячески стремятся следовать. Речь идет о мышах фирм Microsoft, Logitech и Mouse System — они относятся к наиболее популярным изделиям этого рода, к тому же поддерживаемых большинством прикладных программных систем.

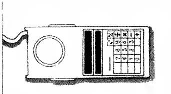
Очень часто мыши поставляются со специальными программами — генераторами меню. Такая программа позволяет пользователю создавать на экране одно или несколько меню и "начинять" их пункты командами управления другими программами. После установки курсора на том или ином пункте и нажатии на клавишу мыши программа направляет соответствующей прикладной программе указанную в меню команду — точно так же, как если бы эта команда была введена с клавиатуры.

Трекбол аналогичен мыши и по принципу действия, и по функциям; различаются они, по существу, только конструктивно. Трекбол представляет собой перевернутую на спину мышь, шар оказывается сверху, и пользователь должен вращать его ладонью или пальцами, а перемещать корпус устройства не надо.

Мышь или трекбол? Иногда ответ совершенно однозначен, но чаще это дело вкуса. И как всегда критерий истины — опыт. Если пока нет своего, то стоит обратиться к чужому. Каковы же мотивы тех пользо-

Трекбол Tracker Mouse

Это устройство рекламируется изготовителем фирмой Penny and Giles как единственный трекбол "на котором можно считать", имея в виду то, что в его корпус встроен калькулятор (запитываемый от солнечной батареи). Никакой функциональной интеграции этого калькулятора с собственно трекболом не просматривается.



В то же время сам трекбол вполне на уровне: удачны пропорции корпуса и шара, хорошо расположены кнопки. Размеры корпуса таковы, что пользователь легко может взять его одной рукой, перемещать шар мягкой частью указательного пальца и, не меняя положения руки, нажимать на фиксирующую выбор кнопку пальцами, охватывающими корпус.

Трекбол Tracker Mouse поставляется с программой KeyBall, которая позволяет использовать его с любой не ориентированной на мышь текстовой программой, но назначение функций кнопок надо программировать самому пользователю, что является достаточно трудоемким делом.

Устройство с успехом применено в медицинской аппаратуре, поставляемой фирмой OSW computer, которая заменила калькулятор цифровым клавишником. Главная причина успеха трекбола в этом применении в том, что медики как огня боятся клавиатуры, а трекбол в модифицированном варианте ее практически заменяет.

Разрешающая способность устройства 200 cpi. Цены модели с последовательным интерфейсом для компьютеров типа PC и PS/2 — 99 долл.

вателей персональных компьютеров (ПК), которые отказываются от такого распространенного устройства управления курсором, как мышь, в пользу шарового манипулятора, трекбола? Приверженцы трекболов приводят следующие три аргумента:

1. Трекбол управляет курсором точнее. И вот почему. Во-первых, даже у самого маленького трекбола диаметр шара больше, чем у мыши, а чем больше шар, тем проще им манипулировать. Во-вторых, ладонью и пальцами удается работать точнее, чем кистью и запястьем, как в случае мыши. Поэтому в применениях, где приходится иметь дело с мелкими деталями, например, в системах автоматизированного проектирования, при разработке электрических схем, трекбол — вещь незаменимая.

2. Трекбол занимает меньше места на столе пользователя. Дело в том, что когда работаешь с трекболом, надо только вращать его шар, а корпус устройства, в отличие от мыши, остается неподвижным. Это свойство делает трекболы идеальным средством для различного рода пультов. Чаще всего речь идет о встраивании трекбола в поверхность моторного поля пульта. В частности, трекболы широко используются авиадиспетчерами для указания объектов на экранах радаров, в автоматизированных системах управления технологическими процессами (АСУ ТП) и т.п. По той же самой причине трекболы должны стать неотъемлемой принадлежностью портативных компьютеров. Последняя тенденция будет определяться тем, как быстро удастся реализовать на машинах этого класса работу с графикой, для которой критически важно управление курсором при помощи манипулятора.

3. Трекбол не нуждается в тщательном уходе. В случае мыши ее обрезиненный шар все время контактирует с поверхностью стола и собирает с нее пыль и грязь, что отрицательно сказывается на работе устройства. Когда же речь идет о трекболе, то до шара дотрагиваются только ладонью и пальцами, поэтому чистить его можно значительно реже.

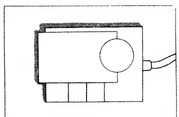
Несмотря на внешние различия трекбол и мышь конструктивно очень похожи — при вращении шар и в том и в другом устройстве приводит во вращение пару колесиков. Одно из них перемещается, когда шар вращается вокруг оси, соответствующей вертикальному смещению курсора по экрану, второе — когда он (шар) вращается вокруг оси, соответствующей горизонтальному смещению курсора. Путем подсчета электрических импульсов, возникающих при повороте колесика на заданный угол (шаг), программное обеспечение, поддерживающее работу трекбола, отображает вращательное движение шара перемещением курсора по экрану.

Мерой главной, коренной характеристики трекбола (как и мыши) является число отсчетов, даваемых упомянутым колесиком на единицу хода шара — cpi (counts Per Inch — число отсчетов на дюйм). Этот параметр определяется количеством зубьев (или других элементов) колесика, при прохождении которых через датчик и формируется выходной импульсный сигнал. Чем больше количество таких элементов на колесике, тем больше величина показателя cpi и, следовательно, выше точность позиционирования курсора.

Изготовители часто характеризуют трекболы другим показателем — числом точек на единицу хода шара —dpi (Dots Per Inch — число точек на дюйм). По названию он совпадает с показателем, характеризующим мониторы и лазерные принтеры, но по существу это не одно и то же. Если для принтера dpi означает

Трекбол Track Man

Этот трекбол фирмы Logitech имеет довольно необычную конструкцию: шар расположен с левой стороны прямоугольного корпуса, а клавишный узел (в нем 3 кнопки) — с правой. Это предполагает, что манипулировать шаром можно только большим пальцем правой руки. На первый взгляд, конструкция представляется удачной: шаром (он имеет сравнительно небольшой диаметр) очень легко управлять (правда, это относится только к работе в среде текстовых процессоров, но тонкого позиционирования курсора, которое необходимо для чертежных и рисовальных пакетов, добиться при таком шаре нельзя), однако когда работать приходится долго, большой палец устает и точность управления курсором теряется. Кроме того, трекбол совершенно не годится для левшей (если не считать тех, у кого очень развиты мизинцы).



Track Man поставляется с программой Mouse Ware, которая выдает меню, позволяющее работать более чем с 24 не ориентированными на мышь программами, и программой Logi Menu, которая дает возможность пользователям адаптировать вышеупомянутые меню или создавать свои собственные. В состав этой программы включена программа Mouse 2-3, выдающая шаблон для пользования системой Lotus 1-2-3 в версиях 1 и 2.

Есть удачный опыт применения этого трекбола с пакетами САПР, текстовыми процессорами. Особенно положительно он проявил себя при работе в рисовальных пакетах, где им легче управлять курсором, чем мышью.

Разрешающая способность устройства 300 cpi. Цены: модель с последовательным интерфейсом — 139 долл.; модель с параллельным интерфейсом — 149 долл.

количество точек, которое устройство может разместить на единице длины (1 дюйм) бумаги, то в случае трекбола речь идет о том, насколько смещается курсор на экране (а смещение измеряется числом экранных точек) при повороте шара на 1 дюйм. Этим показателем и оценивается скорость работы трекбола.

Показатель dpi неоднозначен. Дело в том, что программы обслуживания трекболов могут масштабировать отсчет колесиков. В результате 1-дюймовый ход шара может быть преобразован в смещение курсора, измеряемое числом точек в диапазоне от 50 до 15000 (т.е. от 1/6 до 50 дюймов на экране дисплея типа VGA). На практике большинство программ-драйверов трекболов имеют характеристику, именуемую динамическим ускорением. Это коэффициент, обеспечивающий автоматическое варьирование dpi. Это означает, что при быстром повороте шара курсор пройдет на экране значительно большее расстояние, чем при медленном повороте, даже если ход шара в обоих случаях будет одинаковым.

Конечно, возможность перемещать курсор на большие расстояния, превышающие размеры экрана, представляется совершенно бесполезной. Но фирмы, указывающие соответствующие высокие показатели dpi для своих изделий, имеют в виду, что в этом случае вы можете одним коротким, но быстрым движением шара, переместить курсор на весь экран.

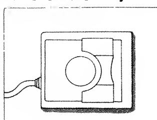
При нормальной работе такие высокие значения dpi роли не играют. Главным показателем трекбола все же остается величина cpi. Именно ею оценивается точность позиционирования курсора. У подавляющего большинства известных моделей трекболов этот показатель находится в пределах 200-300 cpi.

В отличие от мышей трекболы наделяются одной особой функцией — функцией фиксации кнопки, служащей для подтверждения того, что курсор приведен в требуемую точку. Такая кнопка имеется во всех устройствах управления курсором; ее именуют клик-кнопкой — от слова “click“, означающего кратковременное нажатие, нажатие быстрым касанием. Функцию фиксации клик-кнопки называют также функцией управления протяжкой курсора. Наличие этой функции дает вам возможность (в случае мышей это не предусматривается) рисовать, например, линию или перемещать выбранный на экране объект без того, чтобы держать все время нажатой кнопку устройства. Для трекболов это совершенно необходимое свойство, так как нажимать на кнопку и одновременно вращать шар-манипулятор сколь-нибудь длительное время практически невозможно.

Трекболы могут работать с большинством программ, ориентированных на управление мышами (т.е. характеризуются совместимостью в этом аспекте). Для работы с популярными программами, которые не предусматривают применение устройств управления курсором (как, например, первые версии системы Lotus 1-2-3), поставляются специально разработанные драйверы. При этом обычно предоставляется возможность адаптировать работу трекбола (в сравнительно малозначащих деталях) к собственным потребностям. Речь идет, например, об эмуляции (т.е. воспроизведении) работы 1-, 2-, 3-кнопочной мыши или о том, чтобы поменять местами назначение кнопок — чтобы правая выполняла функцию левой и наоборот (последнее может потребоваться, когда трекбол куплен левшой).

Трекбол Pc Trackball

Среди начинающих пользователей, когда они берутся за мышь или трекбол, распространен вопрос: "А мне от этого хуже не станет?" В случае устройства Pc Trackball фирмы Mouse Systems ответ такой: "Да, станет". Дело в том, что его разработчики, по-видимому, считают, что у большинства людей рука имеет либо очень длинный большой и очень короткие остальные пальцы, либо наоборот – большой палец короткий, а остальные очень длинные. Шар удобнее всего вращать вытянутым указательным, пальцем, но при этом большой палец не дотягивается до левой кнопки. Чтобы было удобно нажимать на нее приходится все время работать согнутым указательным пальцем, отчего он быстро устает.



Неудачная конструкция описываемого трекбола не способствует и той тонкой координации движений при работе шаром и кнопками, которая необходима при управлении курсором. Из-за этого нередки случаи, когда пользователь нажимает на кнопку фиксации выбора преждевременно со всеми вытекающими из этого последствиями.

Pc Trackball поставляется с программой-драйвером, которая выдает на экран меню, облегчающее пользование рядом программ, не ориентированных на применение мыши. Кроме того, поставляется пакет графики и рисования Presentation Magician.

Разрешающая способность устройства 200 cpi. Цены: модель с последовательным интерфейсом и модель для компьютеров типа PS/2 – 119 долл.; модель с параллельным интерфейсом – 139 долл.

Сканеры и устройства ОРС

В связи с растущей популярностью систем настольных издательств и широким применением компьютерных (электронных) документов, в которых текстовый материал перемежается графикой (схемами, рисунка-

ми и т.п.), важнейшим средством ввода в компьютер для последующей обработки больших объемов данных и (или) изоинформации становятся сканеры и устройства ОРС. Их привлекательность постоянно растет в силу быстрого падения цен как на серийные изделия этого класса, так и на персональные компьютеры, обеспечивающие эффективное считывание графических изображений.

Сканер изображений и устройство ОРС можно представлять себе как своего рода фотоаппарат, который делает "снимок" того, что изображено на бумаге. Однако после щелчка затвора такого фотоаппарата изображение не попадает на фотопленку, а преобразуется в поток цифровых сигналов. Через интерфейсную плату, установленную в гнезде ввода-вывода, эта изоинформация поступает в компьютер и преобразуется в файл особого формата. Затем этот файл обрабатывается соответствующим программным обеспечением. В отношении графической информации (рисунки, схемы, фотографии), целью обработки является модификация (например, изменение масштаба, ретуширование), а информация, представляющая алфавитно-цифровые символы, преобразуется в текстовую форму, в которой ее можно редактировать, распечатывать и т.п.

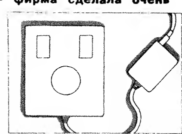
Эти средства целесообразнее всего использовать, когда на компьютере необходимо обработать большой объем информации. В этом случае сканирование позволяет отказаться от необходимости повторно набивать отпечатанный ранее (на пишущей машинке или типографским способом) текстовый материал.

Центральным элементом любой сканирующей системы является датчик, воспринимающий световой поток, который отражается от поверхности бумаги со считываемым материалом. В некоторых сканерах считываемая страница кладется на планшет (так же, как во многих копировальных аппаратах), а в других протягивается через зону восприятия датчика. В любом случае датчик сканирует страницу по всей ее ширине с шагом 1/100 или 1/200 дюйма (1 дюйм = 25.4 мм). В процессе сканирования страница представляется последовательностью строк (линий), каждая из которых составляется из темных и светлых точек-пятен. Эти точки рассматриваются как единичные элементы изображения-пиксели (pixels) и вводятся в компьютер в форме непрерывного потока цифровых (дискретных) сигналов. Поскольку сканирование всегда выполняется стандартизированным способом, то на основе упомянутых сигналов графический образ страницы можно в любое время восстановить с целью, например, отображения на экране или воспроизведения при помощи печатающего устройства. Такой цифровой образ можно также обрабатывать при помощи графики. Например, этим способом можно воспроизводить свою личную подпись под письмами и документами. Для этого ее достаточно просканировать и занести на хранение в компьютер.

ОРС предполагает определенную обработку — преобразование результата сканирования специальной программой, которая ставит в соответствие, составленному из вышеупомянутых точек рисунку алфавитно-цифрового символа, его стандартный компьютерный код. После такого преобразования получается тексто-

Трекбол Trackball

Устройство Trackball фирмы Lynx computer Products может показаться несколько примитивным, но это только внешнее впечатление – фирма сделала очень много, чтобы получилась действительно уникальная вещь. Во-первых, для измерения хода колесиков, на которые передается вращение шара, применены не светодиоды, как это сделано в большинстве трекболов, в своего рода маленькие "щетки", которые воспринимают, на сколько шагов (зубьев) повернулось колесико. Это решение не повлияло на качество работы устройства, но позволило применить более простую электронную схему, в результате фирма смогла пойти на то, чтобы давать гарантию на более длительный, чем другие срок службы устройства.



Еще более полезным новшеством является комплектация трекбола съемным интерфейсным блоком, выполненным в виде корпуса-коробочки, который вставляется (при помощи полуразъемов) в провод, соединяющий трекбол с компьютером. Это дает возможность, меняя этот блок (а он стоит всего 29,95 долл.), использовать один и тот же трекбол с машинами разных типов (фирм Apple и IBM).

Описываемое устройство одинаково удобно и для правшей и для левшей. Только, когда приходится делать длинные протяжки курсора по экрану, возникает ощущение, что лучше было бы это делать не одной, в двумя руками.

Трекбол поступает с программой, выдающей меню для работы в ДОС, с системами Lotus 1-2-3 и Word Perfect. Он получил положительные отзывы ряда пользователей, которые особо отметили точность и стабильность позиционирования курсора, что крайне важно при работе в среде САПР печатных плат, где приходится иметь дело с мелкими деталями.

Разрешающая способность устройства 200 cpi. Цены: модели с последовательным и параллельным интерфейсами – 129 долл.; интерфейсные блоки для компьютеров PC, PS/2, Apple 2 и Macintoch – 29.95 долл. (каждый).

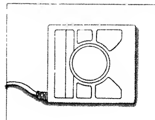
вый файл, с которым могут работать системы обработки текстов (текстовые процессоры), системы типа электронных таблиц и баз данных.

Как разновидность ОРС можно рассматривать факсимильные устройства ввода. Главное их различие в том, что в случае ОРС сканированное изображение поступает в компьютер по кабелю от рядом стоящего сканера, а в факсимильной системе сканер может находиться на большом расстоянии от компьютера и информация поступает по линии телефонной связи.

Еще несколько лет назад сканеры стоили не меньше 25 тыс. долл., а сейчас цены даже на самые высококлассные устройства, например, модели ScanJet фирмы Hewllett-Packard, существенно ниже 2 тыс. долл. Программное обеспечение ОРС поставляется либо вместе со сканерами, либо как отдельный продукт и стоит порядка нескольких сотен долларов. При таких ценах применение систем ОРС экономически вполне оправдывается, когда речь идет о вводе текстов объемом порядка 1000 стр. и больше. В том же случае, когда речь идет о вводе изоинформации, альтернативы сканированию просто нет.

Трекбол Roller Mouse

Этот трекбол (фирма CH Products) отличается большим шаром (размерами с бильярдный) и наличием четырех крупногабаритных кнопок. Но практика показала, что крупные элементы – это еще не все, что надо для того, чтобы получился удобный в пользовании трекбол. Вращать шар кончиками пальцев так же неловко, как пытаться ими касаться внутренней стороны ладони, а если вращать его ладонью, то теряется контакт с кнопками. Правда, последний недостаток несколько сглаживается наличием кнопок с фиксацией (т.е. западающих при нажатии), но такое решение противоречит принципу "безрежимного" действия, согласно которому нежелательно требовать от пользователя выполнения дополнительной операции выбора режима работы устройства. Слишком тугие кнопки и неудачное их размещение приводят к тому, что быстро устает большой палец, которым их приходится нажимать.



Трекбол поставляется с программой, которая выдает меню для работы в ДОС, в среде Lotus 1-2-3 и Word Perfect 4.2. Есть возможность создавать и собственные меню.

Несмотря на все свои недостатки, описываемый трекбол благодаря весьма высокой точности позиционирования курсора, получил довольно хорошую оценку у ряда пользователей, которые накопили опыт работы с ним в среде систем Word Perfect 5.1, Quatro Pro и AutoCAD.

Разрешающая способность: номинальное значение 200 cpi, возможна установка (при помощи переключателей) уровней 100 и 400 cpi. Цены: модель с последовательным интерфейсом и для компьютеров типа PS/2 – 129,95 долл.; модель с параллельным интерфейсом – 149,95 долл.

Другие устройства ввода

Клавиатуры, мыши и трекболы стали, по существу, унифицированными средствами ввода, которые широко используются практически во всех направлениях применений компьютерной техники. Довольно широко в последнее время стали применяться сканеры и системы ОРС. Однако остаются довольно большие категории пользователей, которые работают в условиях, требующих специализированных средств ввода информации в компьютер.

Графические планшеты и диджитайзеры предполагают большую степень участия пользователя в процессе ввода. Это в большой мере устройства механического действия. Графический планшет содержит пластину, на поверхности которой пользователь рисует специальным карандашом. В пластину встроены датчики, которые фиксируют текущее положение острия и передают соответствующий сигнал (предварительно преобразованный в цифровую форму) в компьютер. Последовательность положений острия можно запомнить в памяти компьютера и затем выдать на экран дисплея в виде изображения линии. При этом возможен вариант формирования изображения в режиме реального времени, т.е. практически одновременно и в том же темпе, в каком рисует пользователь.

Другой подход к преобразованию перемещения рисующего инструмента в цифровой код реализуется в планшетном устройстве с движущейся державкой — "рукой". В этом случае датчики непрерывно фиксируют положение этой руки. В процессе рисования (или обводки готового рисунка осуществляется передача в компьютер цифровых кодов — координат точек рисунка.

Третий способ оцифровки изображений или перемещения рисующего инструмента сводится к использованию стандартной видеокамеры, подключаемой к компьютеру через специальную интерфейсную плату, которая размещается в корпусе последнего. В этом случае видеоизображение, последовательность дискретных сигналов, представляющих в стандартизованном формате точки, составляющие снимаемый рисунок, преобразуется специальной программой в стандартизованное цифровое "компьютерное изображение". Полученные изображения можно затем обрабатывать при

помощи обычных графических пакетов прикладного программного обеспечения.

Световые перья и сенсорные экраны считается целесообразным применять в ситуациях, когда клавиатурный ввод слишком труден для пользователя или отнимает у него много времени и когда имеется программное обеспечение, поддерживающее возможность выбора пользователем некоторых вариантов, отображаемых на экране.

Известно много разновидностей технических решений, реализующих принципы светового пера и сенсорного экрана. Одно из них сводится к тому, что перо воспринимает световой поток, излучаемый экраном дисплея. Поскольку каждая точка экрана периодически регенерируется (т.е. подвергается действию электронного луча), компьютер может установить, к какой точке экрана подведено световое перо, определив момент времени, когда перо прореагировало (сигналом на своем выходе) на возникающее при регенерации повышение яркости экрана в точке касания.

Другой подход состоит в том, чтобы создать перед экраном своего рода завесу, сеть из сфокусированных световых лучей. Когда световое перо (или другой объект, например, карандаш) приближается к экрану, происходит прерывание нескольких лучей завесы. Координаты точки касания можно определить по номерам прерванных лучей (по горизонтали и вертикали) с отсчетом их от угловой точки экрана. Далее компьютер определяет, какой элемент текущего изображения на экране имеет данные координаты, и выполняет приписанное этому элементу действие.

Третий подход связан с помещением перед экраном или за ним двух электрически заряженных сеток. При касании экрана пальцем или каким-то предметом происходит сближение этих сеток и формируется электрический сигнал, декодируя который компьютер может определить координаты точки касания. Во всех случаях программное обеспечение компьютера использует полученную информацию о координатах задействованного элемента изображения точно так же, как если бы речь шла о нажатии клавиши, и начинает выполнять запланированное для этой ситуации действие или вариант.

Рынок УВИ, по существу, делится на несколько отдельных секторов, что отражает различную степень популярности и различный спрос на компьютерные устройства ввода. При этом все нарастающая тенденция к оснащению компьютером каждого рабочего места будет означать сохранение большого спроса на клавиатуры. Хотя будет продолжаться рост популярности мышей, и во многих случаях эти устройства станут обязательной принадлежностью компьютеров, все же заменить они клавиатуры не смогут.

Дело в том, что еще не изобретен компьютер, на котором можно было бы работать только мышью, и в то же время большинство компьютеров прекрасно работают только с клавиатурой, без мыши.

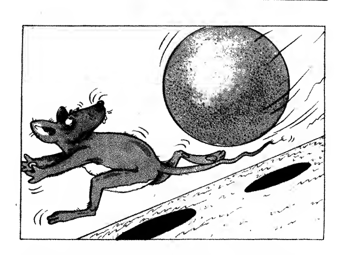
Сканеры и системы ОРС останутся сравнительно специализированными средствами, но потребность в них по мере того, как ввод изоинформации в компьютер будет в глазах пользователей приобретать ту же важную роль, что и представление информации в графической форме на выходе, очевидно, будет возрастать. Графические планшеты и диджитайзеры, световые перья и сенсорные экраны будут оставаться в своих сравнительно узких рыночных нишах. Некоторые из этих устройств станут чрезвычайно важными аппаратными средствами для весьма небольшого количества пользователей, но подавляющему большинству подобного рода экзотика вряд ли потребуется.

В настоящее время цены на клавиатуры и мыши достаточно малы. Хорошую клавиатуру можно купить примерно за 50-150, а мышь — за 35-125 долл. Дальнейшее падение цен представляется маловероятным.

Спрос на сканеры, в том числе и в составе систем ОРС, должен повышаться в связи с совершенством техники работы с графикой и падением цен на модули графических процессоров. Самые высококлассные по быстродействию и разрешающей способности сканеры стоят в розницу больше 10 тыс. долл., но есть вполне приличные модели, которые можно приобрести всего за 1 тыс. долл. и даже дешевле, а ручные сканеры (устройства, в которых сканирующий узел надо перемещать вручную) стоят меньше 500 долл.

Ожидать падения цен на графические планшеты, диджитайзеры, световые перья, сенсорные экраны и другие специализированные УВИ не приходится, поскольку тенденции к увеличению спроса на них не просматривается.

В стадии исследований и конструктивных разработок в настоящее время находится еще ряд новых устройств ввода. Наиболее примечательными из них являются системы распознавания речи (которые можно "научить" слышать и интерпретировать произносимые вслух команды и данные) и т.н. "программные" клавиатуры. Последний подход предусматривает отображение клавиатуры на экране и отказ от аппаратных



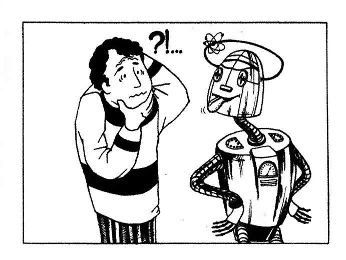
переключателей и клавиш. Разработчики активно пропагандируют эти устройства, добиваясь организации их серийного производства. Утверждается, что они, благодаря своим уникальным свойствам, будут способствовать выходу компьютеров на новый, значительно более высокий уровень гибкости.

Тем не менее не вызывает сомнений, что в ближайшие 5 лет основным средством ввода для компьютеров останутся все те же старые, добрые механические клавиатуры. Их простота, надежность, дешевизна и привычность таковы, что пройдет еще много лет, прежде чем возникнет угроза их доминирующему положению со стороны "конкурентов".

Г.Берг

По материалам:

Faulkner Technical Reports on Microcomputers and Software;
DataPro Reports on Microcomputers.
P.Wallace, All the right mouse. PC Computing, June, 1990.



Десять лет назад был опубликован японский проект вычислительных систем пятого поколения, рассчитанный на реализацию в девяностые годы. Несмотря на то, что проект не предполагал создания какого-либо коммерческого продукта даже к концу 90-х годов, а только проведение фундаментальных исследований, связанных с разработкой новой компьютерной технологии, замысел авторов показался достаточно амбициозным и фантастическим. Сегодня этот проект стал почти реальностью, а отдельные элементы проекта, касающиеся СБИС, компьютеров с параллельной архитектурой и программного обеспечения для них, уже реализованы.

Основные языки программирования искусственного интеллекта

НЕМНОГО ИСТОРИИ

Что видишь ты еще
В пучине темной лет минувших?

У.Шекспир. "Буря"

Основной идеей японского проекта явился отказ от эволюционного подхода к разработке компьютерных систем будущего, состоящего в постепенном совершенствовании существующих технологий, и переход к принципиально новым архитектурам ЭВМ (RISC, CISC, компьютеры с параллельной архитектурой) и технологиям их изготовления (СБИС с количеством элементов более 1 миллиона), что позволило бы сделать следующий шаг в создании новых информационных систем. Разработчики этого проекта считали, что системы обработки информации должны стать важнейшим структурообразующим фактором жизни общества, охватывая экономику, искусство, науку, управление, образование и культуру. ЭВМ пятого поколения должны обеспечить значительный рост качества продукции и производительности труда в промышленности. Японские специалисты считали, что решение этой проблемы по своему значению сравнимо с реализацией продовольственной и энергетической программ. Предполагалось, что с помощью таких систем удастся решить проблемы оптимизации потребления энергии и вообще достичь эффективного использования всех ограниченных ресурсов, оставшихся в распоряжении человечества. Совершенно особая роль отводилась машинам пятого поколения в охране окружающей среды, имеющей определяющее значение для прогресса во всех сферах.

Японскими исследователями была предложена модель ЭВМ пятого поколения, интегрирующая следующие технические и программные средства:

Базовые программные средства
- система решения задач и получения логических выводов
- система управления базами знаний

- система интерфейсов

Программные средства для системотехнических исследований
- система программирования
- система проектирования баз знаний
- система проектирования СБИС

Сервисная система
- система утилит для работы с разнородными базами данных
- средства автоматического контроля и восстановления информации
- справочные средства

Основные базы знаний
- общая база знаний
- системная база знаиий
- базы прикладных зианий

Базовые прикладные системы
- система машинного перевода
- диалоговая система (запрос—ответ)
- система распознавания речи
- система распознавания образов и графических представлений
- система решения прикладных задач

Прикладные системы
- система решения инженерных задач и задач проектирования
- обучающие системы
- автоматизированный офис
- интеллектуальный робот

Довольно неожидаино в этом длинном списке интеллектуальный робот оказался на самом последнем месте. Ведь, казалось бы, создание такого робота и должно являться целью разработки машин пятого поколения, т.е. систем искусствеиного интеллекта (ИИ). Тем не менее, место для этого робота выбрано верное, поскольку путь по его созданию и тяжел, и долог. И если все-таки оглянуться назад, то мы увидим, что со времен первого известного упоминания об искусственном иителлекте до самого последнего времени прогресс в этой области весьма невелик. Первое упомииание об искусственном разуме встречается у Тацита, который пишет, что, согласно легеиде, для того, чтобы защитить Европу от тех, кто попытался бы вновь похитить ее и увезти домой в Фииикию, Зевс создал бронзового робота, задачей которого было потопление всех кораблей, приближавшихся к Криту. Забавна и довольио свежая реальная история о маньяке-компьютере, который работая в банке, регулярно рассылал клиентам уведомления, что за ними числится долг в 00 долларов 00 центов, и не успокаивался до тех пор, пока не получал чек имению иа эту сумму. Не будем критиковать создание Зевса, но вот во втором случае совершенно явно видна ошибка программиста. Старая истина верна: прежде, чем сделать Нечто, надо сделать инструменты, годные для создания этого Нечто. А в нашем случае речь идет о языках программирования искусственного интеллекта, о коих подробнее — ниже.

ТРИ ИСТОЧНИКА И ТРИ СОСТАВНЫХ ЧАСТИ..

По этому поводу я могу строить только расплывчатые и почти не имеющие никакого реального подтверждения догадки. Сравнительная анатомия сделала пока лишь первые шаги, а наблюдения натуралистов слишком неопределенны, чтобы представлять собой основу для серьезных выводов.

Ж.-Ж. Руссо. "Рассуждения о происхождении и основании неравенства между людьми"

Функциональный подход к задачам ИИ (Лисп)

В результате почти полувековой практики программирования был выработан весьма эффективный способ составления программ, при котором задача разбивается на подзадачи, а те в свою очередь дробятся на элементарные функции, определенное сочетание которых реализует решение подзадач и соответственно всей задачи. Процесс дробления на элементарные функции заканчивается тогда, когда каждая из полученных фуикций выполняет некоторую логически отдельную операцию и результат работы данной функции может служить входом для другой функции. Такое разбиение приводит к тому, что задача представляется некоторым минимальным набором функций, который и кодируется иа языке, адекватном характеру задачи.

Описанный функциональный подход оказался весьма эффективным для программирования на языке Лисп — языке обработки списков, ориентированном на символьную обработку даиных. Трудно сказать, что функциональиый подход является привилегией языка Лисп, этот подход дает прекрасные результаты и при использовании языков программирования Паскаль и Си. Первые попытки решить задачи ИИ на ЭВМ были предприняты в американских университетах в конце 50-х годов, и в качестве рабочего был выбран единственно доступный язык символьной обработки — LISP, основанный на алгебре списочных структур, лямбда-исчислении и теории рекурсивных функций. В связи с тем, что американские исследователи были законодателями моды в работах по ИИ, за Лиспом утвердилась слава языка программироваиия для задач ИИ. Более чем за 25 лет своего существования язык Лисп был существенно расширен, появилось множество диалектов, но и по сей день язык Лисп не стандартизован. Наиболее известными диалектами являются INTER_LISP, ZETA_LISP, FRANZ_LISP и COMMON_LISP. Последний можно считать даже неким иеофициальным стандартом.

Осиовными типами данных языка Лисп являются атомы и списки.

АТОМ — это цепочка из любых символов, не содержащая пробелов.

СПИСОК — набор любого количества атомов и/или списков, разделенных пробелами и содержащихся внутри круглых скобок: (15 3 81) (A B (C D (E) F G)).

Для этих типов данных определены операции — функции, являющиеся примитивами языка:

SETQ — присвоение (связывание)

CAR — возврат в качестве значения головной части списка

CDR — возврат в качестве значения ´хвоста´ списка

CONS — включение нового элемента в начало списка

EVAL — вычисление значения выражения, находящегося внутри скобок

DEFUN — определение функций пользователя.

Прочие элементы языка LISP являются результатом добавлений на уровне библиотечных нестандартизованных функций со всеми вытекающими отсюда последствиями.

Эти библиотечные функции позволяют создавать структуры данных пользователя, строить сложные выражения по образцу, использовать макроопределения, варьировать способы задания аргументов функций и осуществлять потоковый ввод/вывод.

Практически все реализации Лиспа требуют наличия мощного процессора и значительных ресурсов памяти. Профессиональная работа на Лиспе возможна в основном на сверхмощных Лисп-машинах типа Simbolics 3600 и FAIM-1 или хотя бы VAX-11/780. Однако имеются реализации Лиспа и для персональных компьютеров.

Фирма Gold Hill Inc. (США) объявила о выпуске транслятора Golden Common Lisp v 4.0 для работы в среде Windows 3.0 фирмы Microsoft. Версия 4.0 обеспечивает динамический обмен данными между Windows и приложениями, разработанными на GC Lisp. Новая версия позволяет осуществлять непосредственный вызов функций, написанных на Си, из GC Lisp-программы. Кроме того, пользователи получили возможность строить собственные динамически связываемые библиотеки. Версия 4.0 выпущена для машин Compaq 386, IBM PS/2 моделей 70 и 80 и совместимых с ними.

Фирма Franz Inc. (США) объявила о выпуске Allegro Common Lisp v 4.0 и версии 2.0 Allegro Composer — многооконной среды для разработки приложений. Эта версия позволяет производить динамическую загрузку функций и использовать дополнительные наборы символов. Эта версия предназначена для рабочих станций, работающих в среде UNIX.

Фирма Lucid Inc.(США) выпустила Common Lisp v.4.0, работающую в среде X-Windows, и Common Lisp Interface Manager (CLIM). Кроме того, фирма поставляет Tool Kit в виде программных средств для повышения эффективности программного кода. Эти программные продукты доступны для рабочих станций Apollo, Sun, DEC, IBM, NCR и Prime.

Фирма Siemens объявила о новой реализации Лиспа для малых машин типа BS2000. Lisp SCL написан на ассемблере. Поставляемая фирмой среда для разработки приложений включает: интерпретатор, компилятор, отладочные средства и системный интерфейс.

Фирма Line System Co Ltd. (Япония) объявила о выпуске транслятора Moebius, который преобразует программы, написанные на Common Lisp´e, в программы на языке Си, которые должны работать в любой UNIX-среде без Lisp-интерпретатора.

Логический подход к задачам ИИ (Пролог)

Мечтой математиков-теоретиков, занятых проблемами ИИ, всегда была возможность применить логику в качестве языка программирования, предназначенного для использования эмпирических знаний и представления закономерностей реального мира. Эти мечты породили в программировании целое направление, называемое логическим программированием, и лишь в малой степени реализовались в виде языка Пролог, изобретенного в 1973 г. группой исследователей Марсельского университета под руководством А. Колмерое.

Логическое программирование должно базироваться на логике первого порядка с ее четко определенной семантикой и синтаксисом. Программист-логик должен записать аксиомы, описывающие задачу. При этом порядок следования аксиом не имеет значения, важна лишь их непротиворечивость. Такой идеал декларативного программирования не удалось реализовать ни группе исследователей Эдинбургского университета, давших в 1979 г. одну из лучших реализаций Пролога на ЭВМ DEC-10, ни другим исследователям. Язык Пролог основан на процедурной реализации логики первого порядка, вызове процедур по образцу и системах описаний. Помимо отдельных устранимых недостатков, Пролог обладает и такими недостатками, с которыми очень трудно бороться, среди которых:

- гипотеза о замкнутости мира, выражающаяся в виде предположения о полноте локально доступного знания
- ограничение логикой первого порядка
- программист должен понимать и постоянно учитывать встроенный механизм возврата при поиске решений.

Гипотеза о полноте локально доступного знания приводит к тому, что некоторое утверждение, если оно не следует из локальной базы знаний, должно быть признано ложным, хотя это вовсе не так в открытой системе. Кроме того, любая система аксиом, выражающая знания о мире, неизбежно оказывается противоречивой. Согласно теореме Геделя, противоречивые системы аксиом бессмысленны, так как из противоречивого набора аксиом можно вывести любое абсурдное утверждение (при достаточном опыте).

Ограничение логикой первого порядка (хорновскими выражениями) не позволяет адекватно описать некоторые задачи, возникающие в реальной жизни. Исследователи делают попытки перейти на логику второго порядка, однако реальных результатов пока нет, да и тогда это уже будет совсем другой язык.

Очень большие надежды возлагаются на параллельные версии языка Пролог, так как они обеспечивают процесс вывода, не зависящий от порядка следования логических предложений и порядка указания целей внутри предложений, и — в силу этого — высокое быстродействие.

Рассмотрим структуры данных и принципы логического программирования на примере наиболее популярного среди советских программистов языке Turbo Prolog фирмы Borland International (США).

ОСНОВНЫЕ ТИПЫ ДАННЫХ Turbo Prolog:
термы
переменные (X,Y,Z,COST)
не-переменные
объекты
symbol
string
integer
real
char
file
списки [1,103,14,21]
структуры NAME(FNAME,LNAME)

СТРУКТУРА ПРОГРАММЫ НА Turbo Prolog:
domains — определение типов данных, используемых в программе
database — предикаты динамической базы данных
predicates — все предикаты, используемые в программе, за исключением динамических
clauses — факты и правила для обработки данных (клозы).

Программа на Turbo Prolog является как бы базой данных, которая неизменна во времени и фактически "заморожена" после того, как программа откомпилирована. Некоторая часть базы данных является динамической и хранится вместе со статической. Динамическая часть базы данных может быть сохранена и восстановлена с помощью предикатов save/consult.

ВСТРОЕННЫЕ ПРЕДИКАТЫ

Turbo Prolog содержит множество встроенных предикатов, выполняющих разнообразные функции по управлению, вводу—выводу данных, работе с экраном, преобразованию типов и т.д.

При выполнении программы встроенный абстрактный интерпретатор Turbo Prolog пытается найти все возможные наборы значений, удовлетворяющие указанной в коде задачи цели. Если программист не включил в тело программы никакой цели, то Turbo Prolog сам запросит ее указания при выполнении программы.

Если пользователь указывает составную цель, то Turbo Prolog обрабатывает последовательно все подцели по порядку слева-направо. Алгоритм работы встроенного абстрактного интерпретатора выглядит так:

а) внутренний маркер устанавливается в начало базы данных;
б) делается попытка удовлетворить самую левую цель в дереве подцелей
в) осуществляется проверка, удовлетворяется выбранная подцель или нет; если удовлетворяется — переход на "г)"; если не удовлетворяется, осуществляется проверка, имеются ли еще подцели в дереве; если таковые имеются, возобновляется просмотр базы данных с текущего положения маркера; если в дереве подцелей больше ничего нет, то выдается сообщение, что цель не удовлетворена;
г) маркер перемещается соответственно удовлетворенной цели; производится присвоение соответствующих переменных;
д) осуществляется поиск подцелей справа в дереве подцелей; если таковые имеются, то следует сообщение о найденном решении; если таких подцелей нет — переход на "е)";
е) устраняется подцель и осуществляется рекурсия с сокращенным списком подцелей.

Естественно, что далеко не все задачи требуют такого стандартного алгоритма обработки дерева подцелей. Для этого пользователю дается возможность управлять обработкой подцелей с помощью предиката '!', не имеющего аргументов, который позволяет прервать стандартный алгоритм обработки подцелей при реализации указываемых программистом условий.

Несмотря на отмеченные выше недостатки Пролога как языка логического программирования, он позволяет довольно быстро сделать работающий прототип системы и поэкспериментировать с ним. При этом необходимо тщательно проанализировать механизм обработки целей и, если это требуется, отрегулировать его с помощью расстановки предиката '!'. Если пользователя не удовлетворяют временные характеристики работы прототипа системы, можно переписать наиболее критичные куски программы на Си.

В настоящее время имеется множество реализаций Пролога. Будет интересно проследить основные тенденции в последних версиях программных средств, поступивших на рынок программного обеспечения.

Фирма Quintus Computer Systems Inc. (США) объявила о выпуске версии Quintus Prolog 3.0, позволяющей включить программы, написанные на языке PROLOG, в приложения, написанные на языке Си. Эта версия (цена 10000 долларов) работает в среде X-Windows и доступна для рабочих станций Sun-3 и Sun-4.

Фирма Elsa Software (Франция) объявила о выпуске объектно-ориентированной системы, написанной на Прологе. Согласно утверждению фирмы, это полностью декларативный объектно-ориентированный язык. От Пролога унаследована концепция не-детерминизма, принцип обработки дерева целей и сравнение с образцом, а от объектно-ориентированного программирования взяты объекты и механизм наследования. Система работает в интерактивной многооконной среде Lap-Media.

Фирма Prologia (Франция) объявила о выпуске Prolog-III. Эта версия позволяет определять различные

ограничения для клозов в виде численных, булевских и других выражений. Эта версия предлагается для решения сложных задач, таких как планирование производства, инженерное проектирование и задач из области финансов.

Фирма Expert Systems Ltd. (Англия) объявила о выпуске Prolog-2 для среды Windows 3.0 фирмы Microsoft. Этот программный продукт использует полную 32-битную адресацию процессора i386 и систему виртуальной памяти. Prolog-2 сосуществует со средой Windows и отличается высокой производительностью на машинах, оснащенных процессором 80486 с тактовой частотой 33 МГц. Цена лицензии 1495 ф.ст.

Фирма Paralogic Inc. (США) объявила о параллельной реализации эдинбургского Пролога для транспьютеров. Программное обеспечение, называемое n-parallel Prolog, автоматически распределяет Пролог-программу по n транспьютерам. При этом программа не должна модифицироваться для работы на различном количестве транспьютеров. В настоящее время выпущен интерпретатор, который работает с Inmos-сопроцессорами, а также доступна платформа, работающая на сети IBM PC без транспьютеров. В качестве ядра системы предполагается использование 32-разрядного RISC-процессора, работающего в параллели с другими транспьютерами. Стоимость поставки программного обеспечения для одного транспьютера составляет 250 долларов за интерпретатор и 10250 долларов за n-parallel Prolog и платформу для пяти транспьютеров.

Объектно-ориентированный подход к ИИ (С + +)

Читатели, проявляющие хотя бы некоторый интерес к зарубежной периодике, могут отметить стремление разработчиков программного обеспечения работать в объектно-ориентированной среде. Даже разработчики программ на Лиспе и Прологе стараются дописать стандартное матобеспечение, чтобы иметь возможность работать в объектно-ориентированной идеологии. И это не просто мода, а естественное стремление сократить время разработки, облегчить повторное использование отлаженных модулей и снизить издержки на сопровождение работающего матобеспечения. Объектно-ориентированое программирование (ООП) — это новая философия программирования, новый способ мышления для программистов, самым существенным моментом которого является объединение структур данных с операциями, которые можно осуществлять с этими структурами данных. Эта новая идеология является развитием старого хорошо известного структурного подхода, она в большей степени соответствует мышлению нормального человека (непрограммиста). После того как произведен структурный анализ проблемы и выделены необходимые для описания проблемы структуры данных, аналитик уже должен для себя представлять, какие операции он может осуществлять с выбранными структурами данных.

Мышление человека имеет скорее ассоциативный характер, и при рассмотрении новых проблем мы пытаемся связать возникающие новые концепции с уже известными и установить между ними дедуктивные связи. Весьма продуктивными оказываются попытки классифицировать проблему и представить ее в виде дерева, на нижних уровнях которого находятся уже известные структуры, а верхние уровни являются обобщением концепций и структур нижних (исходных) уровней. Именно так человек постигает окружающий его мир с древних времен (уже китайская и древнегреческая философия ввели исходные элементы для объяснения мира). Поэтому нет ничего удивительного в том, что именно исследователи в области ИИ первыми взяли на вооружение ООП. Программирование имеет полувековую традицию, основанную на идеях, заложенных фон Нейманом, и выработало свою идеологию, выраженную в таких языках программирования, как Ассемблер, Фортран, Паскаль, Ада, Си. Все эти языки по своей структуре "привязаны" в большей или меньшей степени непосредственно к устройству машины. Новый язык программирования С++ в первую очередь "привязан" к процессу мышления человека и лишь на нижних уровнях — к устройству ЭВМ. ООП позволяет по- новому взглянуть на процесс программирования вообще и на составление программ в частности. Естественно, язык программирования не родился вдруг на пустом месте, ему предшествовали языки SIMULA и SMALLTALK, которые в качестве данных использовали "объекты". Именно с этими языками впервые было связано понятие ООП. Они не получили широкого распространения, и лишь в 1985 году с выпуском спецификаций объектно-ориентированного языка С++ корпорацией AT&T (Bell Laboratories) стало ясно, что в программировании грядет революция. Будем надеяться, что эта революция не принесет трагических последствий для человечества, так как никто пока не собирается отказываться от сделанного в программировании за 50 лет.

Центральным понятием в языке С++ является понятие класса (class), являющееся обобщением понятия структуры в стандартном языке Си.

ОБЪЕКТ имеет уникальный набор переменных, который соответствует по имени и типу элементам данных, определенных для его класса.

УКАЗАТЕЛЬ (pointer) на объект обеспечивает косвенный способ доступа к объектам. Определив набор классов и операций, мы можем понять, что конкретно делает данная программа. Именно в определении класса мы используем основной принцип ООП — инкапсуляцию.

ИНКАПСУЛЯЦИЯ — объединение в одном элементе и данных, и процедуры их обработки. Именно инкапсуляция делает С++ таким привлекательным языком для программирования задач ИИ, так как мы можем определить данные, входящие в классы, и действия, которые могут выполняться над этими данными, как некоторую структуру-объект в системе,

работающей согласно набору правил (продукционной системе), или определить объекты, соответствующие фреймам, и обращаться к ним в программе как к объекту.

Вторым основополагающим принципом ООП является принцип наследования.

НАСЛЕДОВАНИЕ — это сохранение, перенос атрибутов данных и выполняемых над ними операций от объекта к объекту. С помощью этого принципа строятся различные иерархии классов (простое наследование), а также смешанные классы (множественное наследование), когда некоторый новый класс одновременно наследует атрибуты и выполняемые над ним операции от нескольких базовых классов. При этом имеется возможность модифицировать "поведение" объектов. Аналогия с генетикой в данном случае весьма яркая.

ПОЛИМОРФИЗМ означает возможность единообразного обращения к объектам в тексте программы при сохранении уникальности поведения объектов. Этот принцип позволяет определять целый ряд объектов на основе одного базового класса и обращаться к ним единообразно при сохранении специфического поведения каждого из объектов. Так операция сравнения различных объектов возможна только тогда, когда базовый класс определяет метод сравнения.

Именно эти три основных принципа ООП: инкапсуляция—наследование—полиморфизм делают ООП на языке C++ таким многообещающим. Возможно, некоторые пользователи испытывают неприятные впечатления от нового синтаксиса C++, отличающегося от привычного синтаксиса стандартного языка Си, и совсем другой философии программирования. Как только вы освоитесь с новой идеологией, все остальные проблемы покажутся вам не такими уж серьезными.

Фирма Zortec Inc.(США) объявила о создании пакета C++ Developer Edition v. 2.1 для MS-DOS, Windows 3.0, OS /2, DOS 386, UNIX 386. В DOS 386 имеется возможность адресации до 4 Гигабайтов памяти. Кроме того, фирма объявила о выпуске библиотеки классов C++ Database для создания объектно-ориентированных баз данных. Эти программные продукты имеют возможность работы с Virtual Code Manager (VCM) — системой управления виртуальной памятью, которая позволяет использовать до 4 Мегабайтов памяти в программе пользователя. Транслятор C++ может генерировать код для всех указанных операционных систем.

Фирма Borland International выпустила два объектно-ориентированных программных продукта: Turbo Pascal v.6.0 и Turbo C++ v.1.0. Turbo C++ соответствует стандарту корпорации AT&T v.2.0. Оба программных продукта имеют дружественную к пользователю интерактивную многооконную среду для разработки программ (IDE) и встроенную систему управления виртуальной памятью, управление которой доступно пользователю из программ (VROOMM). Кроме того, фирма поставляет Turbo C++ Professional для профессиональных разработчиков. Поставка включает транслятор C++, реализующий два стандарта ANSI C и стандарт AT&TC++. В эту поставку входит также символьный отладчик Turbo Debugger, Turbo Assembler, полностью совместимый с ассемблером MASM фирмы Microsoft и ProFile, позволяющий оптимизировать программный код.

А. Пантелеймонов

Литература:

1. ЭВМ пятого поколения. Концепции, проблемы, перспективы/ Под ред. Мота-Ока Т. М.: Финансы и статистика, 1984.
2. К.Фути. К вычислительным системам пятого поколения/ Язык Пролог в пятом поколении ЭВМ. М.: Мир, 1988.
3. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. М.: Мир, 1990.
4. Berk A. LISP the Language of Artificial Intelligence PrenticeHall, 1988.
5. Уинстон П.Г. ЛИСП совершает революцию/ Реальности и прогнозы искусственного ителлекта. М.: Мир, 1987.
6. Expert Systems, v.7, No. 4, Nov., 1990.
7. Turbo PROLOG Reference Manual. Borland Int.
8. Ladd S.R. Turbo C++. Tecniques and Application. M&T Publishing, Inc., 1990.
9. BYTE, Fall 1990 IBM Special Edition.

ОСНОВНЫЕ ПОНЯТИЯ, ИСПОЛЬЗУЕМЫЕ ПРИ ПРОГРАММИРОВАНИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ОБЪЕКТЫ — элементы, посредством которых описывается исследуемая система (универсум).
Объекты бывают:
- элементарные
- составные
- объекты-переменные.

ОТНОШЕНИЯ — множество логически связанных объектов.

МИР — набор знаний, который может использоваться при решении той или иной задачи. Одновременно в универсуме могут сосуществовать несколько миров. Мир может создаваться динамически из некоторого множества объектов.

КЛАСС — это множество объектов с идентичными свойствами. Свойства объектов класса задаются структурой данных объектов (схемой экземпляров класса) и определенными на них отношениями.

ФАКТ — это констатация того, что между объектами выполняется определенное отношение. Факт является простейшим видом утверждения.

ЦЕЛЬ — объект, являющийся предметом поиска в базе данных.

ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ служат для обозначения неопределенных объектов. В логических программах переменная обозначает неопределенный, но единственный объект, а не некоторую область памяти, как в программах на языках Бейсик, Фортран, Паскаль, Си и Лисп. Логические переменные обозначаются заглавными буквами.

ТЕРМ — это базовая структура данных в логических программах (константы и переменные).

ФУНКТОР — запись операции, выполняемой над термами. Функтор задается именем и своей арностью (числом аргументов).

СОСТАВНОЙ ТЕРМ — структура данных, содержащая функтор и последовательность из одного или более аргументов, являющихся термами.

ВЫЧИСЛЕНИЕ логической программы P состоит в построении примера, логически выводимого из программы P. Цель G выводима из P, если существует такой пример A цели G, что A <-B1, B2, ..., Bn, n>0 — пример предложения в P, где каждое B выводимо из P.

ПРАВИЛА — это утверждения вида A<—B1, B2, ..., Bn, где n>0. При этом A называется заголовком правила, а B1, B2, ..., Bn — телом правила.

ПРОЦЕДУРА — совокупность правил с одним и тем же утверждением в заголовке.

ЛОГИЧЕСКАЯ ПРОГРАММА — это конечное множество правил (предложений).

ВОПРОС — конъюнкция вида A1, A2, ..., An?, где n>0 и A1,...,An — это цели. Считается, что переменные в вопросе связаны квантором существования.

РЕЗОЛЬВЕНТА — текущая цель на некоторой стадии вычисления логической программы.

ДЕРЕВО ВЫВОДА состоит из вершин и ребер и изображает цели, снимаемые в процессе вычисления логической программы.

В августовском номере журнала The Futurist, органе всемирного общества футурологов, представитель фирмы Interleaf Дэйв Уайнбергер пишет, что будущие документы будут не только предоставлять информацию, но и подстраивать свое содержимое под конкретного читателя.

По Уайнбергеру, компьютерные документы скоро будут сами добавлять информацию, изменять графику и даже определять, что читатель может, а что не может увидеть. Это будут своеобразные "активные публикации", меняющиеся в зависимости от того, кто их читает.

Эта идея возникла не сегодня. На осенней выставке Comdex, демонстрируя подобную систему, Уайнбергер заявил, что система "умных документов" требует для своей работы быстрого компьютера на 80386 процессоре или Макинтоша.

Newsbytes News Network, 23 July, 1991

Исследование, проведенное профессором Джеймсом Шиди из Калифорнии показало, что работоспособность человека, использующего 19-дюймовый черно-белый монитор с тактовой частотой 67 Гц и отображением черных букв на белом фоне на треть выше работающего на стандартном VGA-мониторе.

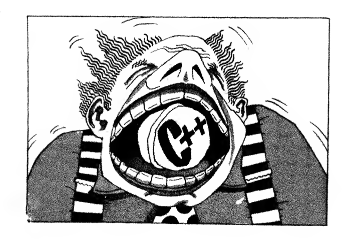
Исследование было проведено по заказу фирмы Cornerstone Technology, которая и производит эти хорошие мониторы (2495 долларов - больше, чем цена многих компьютеров).

Проверялась способность человека в течение длительного времени читать один и тот же текст на экране. При этом не было получено никаких свидетельств того, что частая смена документов или перевод взгляда с экрана на бумагу, как это бывает при вводе информации, может улучшить производительность труда.

Общая тенденция состоит в том, что белый фон на экране лучше, чем черный, высокая частота развертки монитора лучше, чем низкая, и легче читать тексты с более высоким разрешением.

Еще одно исследование, произведенное калифорнийским медиком, продемонстрировало существенно более высокую продуктивность при чтении текста на бумаге по сравнению с компьютерным экраном.

Newsbytes News Network, 23 July, 1991



Предлагаемый материал предназначен для программистов, знакомых с языком С. Статья последовательно вводит в язык С++. Синтаксис языка и его основные идеи представлены достаточно полно, что позволяет совершить легкий переход к новому языку. Тем не менее статью нельзя рассматривать как описание языка С++ — это скорее попытка поделиться двухлетним опытом работы с системами Turbo C и Turbo C++.

От С к С++. Записки хакера

Развитие технологии и языков программирования

Существующее разнообразие языков программирования легко поставит в затруднение человека неискушенного. Когда начинающий программист приступает к работе над новым программным проектом, ему бывает сложно выбрать подходящий язык в том море инструментальных пакетов, где BASIC и FORTRAN соседствуют с С++, объектными MODULA и PASCAL. Этот выбор станет проще, если понять развитие идей программирования и их реализацию в языках программирования. Отсутствие ясного понимания задач и целей, которые преследовались при создании того или иного языка, вызывает споры и попытки сравнения часто несравнимых языков. Видимо, отсюда появляются журнальные статьи-монстры типа "Структурное программирование на BASIC". Только поняв идеи, заложенные авторами в тот или иной язык, можно использовать его в полную силу. Ниже мы попытаемся проследить развитие концепций программирования и проанализировать, как они влияли на вновь создаваемые языки.

Историю языков программирования высокого уровня традиционно ведут с появления языка FORTRAN. Действительно, FORTRAN стал первым языком, для которого был реализован транслятор. Современные поклонники этого языка вряд ли узнали бы синтаксис первых версий FORTRAN. Достаточно сказать, что он не предусматривал такого понятия, как подпрограмма, ведь язык был предназначен исключительно для трансляции формул (FORmula-TRANSlator). Однако этот древнейший предок современных языков уже содержал в себе концепцию ЯЗЫКА ВЫСОКОГО УРОВНЯ, позволившую программисту отвлечься от жесткой привязанности к машине, памяти, адресам и заняться созданием алгоритма, а не чисто механическим переводом своих идей в машинные коды.

Совершенствование синтаксиса языка часто выпадает из поля зрения, когда речь заходит о развитии концепций программирования. Да и самих этих концепций иной специалист назовет не больше одной-двух. Между тем FORTRAN вобрал в себя многие идеи, вся красота и мощь которых стала ясна значительно позже. Одна из них — идея подпрограммы. Как уже было сказано, древнейший FORTRAN не знал понятия подпрограммы; она появилась в нем позже, предвосхитив идею МОДУЛЬНОГО ПРОГРАММИРОВАНИЯ.

Следующим этапом в развитии технологии программирования стала концепция СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ. Часто при попытке описать суть

этой концепции говорят лишь о том, что она запрещает использование оператора безусловного перехода GOTO. Но это не совсем так. Данный подход содержит в себе несколько идей, позволяющих представлять алгоритмы в виде блочной структуры, а не при помощи нагромождения операторов GOTO. Эти идеи определили и синтаксис нового языка программирования — ALGOL. Назовем отличительные черты этого языка:

- операторные скобки, синтаксически объединяющие группу операторов в один оператор-блок;
- условный оператор с альтернативой (IF-THEN-ELSE);
- несколько разновидностей операторов цикла (индексный, с предусловием, с постусловием, по перечислению, безусловный с выходом непосредственно из тела цикла).

Все эти нововведения позволили отказаться от применения оператора GOTO — он просто стал не нужен. Кроме этого, появилось требование ОБЯЗАТЕЛЬНОГО ОПИСАНИЯ ДАННЫХ. Так, была предпринята попытка систематизировать труд программиста; сделать его педантом, аккуратно описывающим данные и алгоритм.

Если удалось структурировать алгоритм, то почему бы не структурировать и данные? Эта идея была заложена в концепцию АБСТРАКЦИИ ДАННЫХ. Смысл ее состоял в том, чтобы позволить программисту описывать новые типы данных со сколь угодно сложной структурой и работать с ними как с единым целым. Такой подход был немедленно использован в языке PASCAL. Программисты ALGOL хорошо представляют себе все ухищрения, к которым надо было прибегнуть при попытке работать со временем, представленным в виде часов, минут и секунд. На PASCAL же достаточно описать новый тип переменных time, представляющий собой запись (структуру), содержащую три поля: hour, min, sec. Программисты, сразу начавшие на PASCAL или другом подобном языке, просто не замечают всего значения такого подхода. Конечно, приведенный пример прост; его легко реализовать и не имея специальных средств. Но представьте себе, что вам надо работать с объектами более сложной структуры, содержащими десятки полей данных. Как быть тогда? Здесь на помощь приходят языки типа PASCAL.

С течением времени программные проекты становились сложнее и больше по размеру. Развитие технологии программирования требовало подхода, который позволил бы выделять алгоритмы, данные и описания новых типов в независимые блоки — МОДУЛИ. Модульный подход заключается отнюдь не в разбиении программы на части. В языках, ориентированных на модульный подход, предусмотрены специальные средства, позволяющие создавать независимые пакеты (под)программ, устанавливать жесткий контроль типов, определять **видимость** переменных и процедур. В таких модулях различают РАЗДЕЛ ОПИСАНИЙ и РАЗДЕЛ РЕАЛИЗАЦИЙ. Раздел реализаций содержит информацию, необходимую для помещения данного модуля в библиотеку (алгоритмы и данные). Раздел описаний содержит информацию, необходимую транслятору для правильной связи (интерфейса) с данным модулем (описания типов, данных, типов процедур и их формальных параметров). При "сборке" головной программы раздел реализаций не транслируется — транслятор всю необходимую информацию получает из раздела описаний, а процедуры берутся уже готовыми из библиотеки. Всеми этими чертами обладает MODULA-2, С и некоторые реализации PASCAL (Turbo 4.0 и выше).

Новомодной концепцией стало ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. Данный подход заключается в объединении данных и алгоритмов, относящихся к одному типу объектов, в единое описание КЛАССА ОБЪЕКТОВ (ИНКАПСУЛЯЦИЯ). При этом алгоритмы называются МЕТОДАМИ данного класса. Кроме того, в объектно-ориентированных языках предусмотрен механизм ПОРОЖДЕНИЯ КЛАССОВ — когда порождаемый класс НАСЛЕДУЕТ данные и методы порождающего класса. При этом имеет место ПОЛИМОРФИЗМ методов — метод с одним именем может исполняться по-разному для порождаемого и порождающего классов (допускаются различные реализации). Разумеется, вы можете добавить в порождаемый класс новые данные и методы. Как правило, предусматривается способ описания доступности полей данных и методов. Объектно-ориентированными языками являются С++ и некоторые реализации PASCAL (Turbo *5.5* и выше).

Объектно-ориентированный подход предоставляет не только удобный синтаксис языка. Стержнем его следует считать механизм порождения классов. Возможность создания разветвленной иерархии классов позволяет сблизить программиста-профессионала и специалиста, занимающегося программированием для решения своих прикладных задач. Первый занимается конструированием иерархии классов, описывая их методы и поля данных, а второй — написанием программы в терминах, близких его специализации (используя описания классов).

Прочитав все это, кто-то скажет: "Да я вам любую программу напишу на FORTRAN'e!". Добавим, что и в машинных кодах можно написать любую программу (теоретически), но так ведь никто не поступает. Все дело во времени и усилиях, затраченных на создание и отладку программы. Любой язык программирования вбирает в себя современные ему идеи, систематизирующие и облегчающие труд программиста. Это один из главных аргументов при выборе подходящего языка!

Модульность

1. Техника перевода. Транслятор, редактор связей

Turbo C++, как и прочие его Turbo-предшественники, придерживается традиционной технологии перевода программ в исполняемые коды. Эта технология, наработанная поколениями программистов, применяется в подавляющем большинстве трансляторов.

Процесс перевода программы в исполняемый файл делится на два этапа: трансляция и сборка (компоновка или редактирование связей). Трансляция заключается в переводе программы, написанной на языке программирования, в исполняемый код. Однако при этом транслятор не в состоянии указать конкретных адресов внешних имен; он просто заменяет их соответствующими ссылками. Что это значит? Предположим, у вас есть функция func1(), и вы вызываете из нее функцию func2(). Транслятор понятия не имеет о том, где находится func2(), поэтому при вызове он не в состоянии указать конкретный адрес этой функции (да это и не входит в его задачи). Транслятор просто составляет из имен функций таблицу внешних имен или ссылок. В эту же таблицу попадают и внешние переменные (не путать с глобальными). Таблица внешних имен и ссылок в совокупности с исполняемым кодом и составляют файл с расширением *.obj, или объектный файл. Строго говоря, создаются две таблицы: одна содержит список имен и адресов объектов (функций и переменных), которые присутствуют в данном файле, другая — список имен, на которые были сделаны ссылки и адреса которых надо конкретизировать.

Теперь наступает очередь редактора связей (компоновщика). В его задачи входит сборка объектных файлов (их, как правило, несколько) и разрешение внешних ссылок (подстановка конкретных адресов), указанных в соответствующих таблицах. Упрощенно это выглядит так: все исполняемые коды записываются подряд, после чего расставляются уже конкретные адреса функций и внешних переменных. Однако на этом работа обычно не заканчивается. Редкая программа обходится без обращения к библиотекам (так, например, функция printf() помещается в одной из системных библиотек). Поэтому остается еще масса неудовлетворенных внешних ссылок, которые компоновщик пытается удовлетворить, отыскивая соответствующую функцию в библиотеке. Если функция найдена, то ее исполняемый код добавляется в конец программы; если нет — то внешняя ссылка остается неудовлетворенной (или неразрешенной). Если удовлетворены все внешние ссылки, то создается исполняемый файл. Если нет — выдается сообщение об ошибках.

Библиотеки отличаются от объектных файлов лишь тем, что содержимое объектных файлов полностью попадает в исполняемый файл, а из библиотек выбираются только необходимые фрагменты. Поэтому наиболее часто используемые функции следует объединять в библиотеки.

2. Оформление модулей

Предположим, что вы объединили часть своих функций в модуль и хотите воспользоваться преимуществами концепции модульного программирования. Для этого вам придется разбить свой модуль на две части — раздел описаний и раздел реализаций. Раздел описаний содержит в себе все описания типов, переменных и функций. Раздел реализаций содержит тексты самих функций. (Более подробно см. в разделе “Развитие технологии и языков программирования“.)

В языке С традиционно оба раздела помещаются в разные файлы. Раздел реализаций помещается в файл с расширением *.С, а раздел описаний — в файл с расширением *.Н (Н-файл, или header-файл). (Сделаем оговорку, что все приведенные ниже примеры программ будут правильно работать в системе Turbo C++, однако с другими трансляторами могут возникнуть некоторые проблемы.)

Допустим, ваш модуль называется MOD1 и содержит функции func1(), func2(), func3(). В этом случае содержимое файлов может быть следующим:

```
/* Файл MOD1.H */

#ifndef __MOD1__H
#define __MOD1__H

int func1 (int x);
double func2 (int x, int y);
double func3 (double z);

#endif
/* конец */
```

Операторы препроцессора заставляют транслятор обходить данный файл в том случае, если он уже транслировался.

```
/* Файл MOD1.C */

#include "MOD1.H"

int func1 (int x) {
/*
............. Текст функции func1()
*/
};
double func2 (int x,int y){
/*
............. Текст функции func2()
*/
};
double func3 (double z){
/*
............. Текст функции func3()
*/
};
/* конец */
```

Если теперь вы захотите использовать функции из модуля MOD1, то вам достаточно будет сделать доступным транслятору файл MOD1.H при помощи оператора #include, а задачу сборки решит компоновщик.

Теперь предположим, что вы написали несколько модулей (один из которых содержит функцию main()) и хотите собрать их в единую программу. Здесь у вас есть выбор. Возможность номер один: воспользуйтесь файлом проекта. В этом файле вы перечисляете модули, входящие в проект (возможно, написанные на разных языках программирования). Интегрированная среда Turbo C будет автоматически транслировать модули проекта в объектные файлы и передавать их компоновщику. Возможность номер два: поместите ваш модуль в библиотеку. В этом случае надо описать

вашу библиотеку в файле проекта. Первый вариант предпочтительнее на этапе отладки модуля, второй — если модуль уже отлажен.

Есть и третий путь — включить файл реализаций в головной файл, используя оператор #include.

Простые расширения языка C

1. Новое в описании типов и переменных

Надо сказать, что создатели языка C++ постарались учесть критику сторонников жестко типизированных языков. Язык стал более требовательным к соответствию типов параметров функций и возвращаемых значений. Теперь любая функция, используемая в программе, должна быть описана до первого к ней обращения. Жесткое ограничение наложено и на типы указателей. Даже для переменной типа void необходимо задавать явное преобразование типа. Например, размещение динамического массива типа double должно выглядеть так:

```
double *a;
...............
a = (double *) malloc (N * sizeof(double));
```

Упрощено введение новых типов. В языке C для ввода нового типа нужно было использовать слово typedef. Теперь любое описаиие структуры или совмещения (union) — его еще называют объединением — вводит новый тип. Если вы описали структуру или совмещение, то вы с полным основанием можете использовать это имя при описании новых переменных, опуская слово struct или union.

Переменная может быть описана в любом месте блока, заключенного в фигурные скобки. Раньше переменная описывалась до первого исполнимого оператора в блоке, теперь это ограничение снято. Главное, чтобы переменная была описана до ее использования.

В язык введено понятие констант. Если раньше иадо было писать:

```
#define PI 3.1415
```

то теперь допускается следующая запись:

```
const PI = 3.1415
```

При этом транслятор сам определит тип константы и будет следить за тем, чтобы ее значение не менялось.

После введения этих соглашений язык C++ стал более строгим, но также и более стройным в отношении работы с типами.

2. Комментирование конца строки

В C++ добавлен новый способ записи комментариев. Вы можете закомментировать конец строки, отделив его двумя символами '/'. Конец строки вместе с этими символами не будет воспринят транслятором.

3. Функции inline

Любой программист на C наверняка сталкивался с побочными эффектами при применении макроязыка. Новичок приходит в восторг, обнаружив, что при использовании макроподстановки isdigit (getc (file)) вводится два символа. Такое проявление побочных эффектов связано с многократным использованием аргументов в макроопределении #define. Основной силой этого оператора считалось то, что он создает как бы функцию, код которой вставляется непосредственно в место вызова. Однако наличие побочных эффектов значительно ограничивало его применение.

Все проблемы разрешила функция с описателем inline. Эта идея была применена в языке ADA и позднее использована создателями C++. При обращеиии к функции с описателем inline не происходит реального вызова, а исполняется код функции, который транслятор вставляет непосредственно в место обращения. Это позволяет экономить время на вызове функции. Естественно, что функция должна быть достаточно короткой.

Вот пример такой функции:

```
inline double sqr (double x) {return x*x;}
```

Надо сказать, что реализация этой функции должна быть "под рукой" у транслятора — иначе он не сможет вставить ее код в нужное место. Поэтому описание таких функции надо помещать не в раздел реализаций, а в раздел описаний (в header-файл — файл с расширением *.H).

Функции, содержащие циклы или ассемблерные вставки, не могут быть inline-функциями. Транслятор проигнорирует этот описатель, выдав соответствующее сообщение.

4. Функции overload (перезагружаемые)

Важным расширением, также пришедшим из языка ADA, является то, что транслятор C++ различает функции не только по именам, но и по типу аргументов. Например, функцию

```
double sqr (double x) {return x*x;}
```

можно дополнить функцией

```
int sqr (int x) {return x*x;}
```

Обе спокойно уживаются в одной программе. При вызове будет использована одна из функций. Какая — решает тип аргумента.

Раскроем секрет транслятора Turbo C++. Историческое название overload никакого отношения к действительности не имеет. Просто транслятор в своей работе использует внутренние имена функций, существенно отличающиеся от используемых в программе. Эти имена содержат в себе скрытое описание типов аргументов. Разные типы аргументов — разные имена. С этими же именами работают программы компоновщика и библиотекаря. Данный механизм и позволяет

нам иметь несколько разных функций с одинаковыми именами, но с разными типами аргументов.

Заметим, что транслятор не различает функции по типу возвращаемого значения.

5. Параметры функции по умолчанию

Допустим, вы имеете функцию для вывода матрицы в файл. Как правило, такие функции снабжаются сервисным набором "бантиков" и "финтифлюшек", которые используются редко или единообразно. Например, наша функция позволяет задать формат вывода чисел и заголовок вывода. При этом нежелательно заставлять программиста указывать все необходимые параметры при вызове функции. Не вдаваясь в описание типа MATRIX и реализацию самой функции, приведем ее описание:

```
print (MATRIX A, char *format = "%9.4 ", char *title = NULL);
```

После этого законны следующие обращения к функции:

```
MATRIX C;
.....................
print (C);
print (C, " %12.7e ");
print (C, " %12.7e ", "-- Матрица C --");
```

Если для некоторых аргументов функции можно указать наиболее часто используемые значения, то C++ позволяет это сделать, избавляя программиста от необходимости указывать их каждый раз при обращении. При описании функции вы можете задать значения "по умолчанию" для нескольких последних аргументов. Эти значения будут использованы при вызове функции в случае, если соответствующие параметры окажутся опущенными. Для этого после имени аргумента через символ '=' указывается его значение.

Не забывайте, что здесь возможны конфликты при использовании механизма overload.

Задать значения параметров можно только один раз, и лучше всего это сделать в разделе описаний.

6. Ссылки

В язык введен новый вид переменных — ссылки. Ссылка — это переменная, задаваемая указателем. Чтобы сделать переменную ссылкой, необходимо после описателя типа поставить оператор "&". Ссылка во всем ведет себя так же, как и переменная того же типа, но при этом надо помнить, что на самом деле она совпадает с другой переменной, адрес которой указывается при объявлении ссылки.

Вот пример:

```
int a;
int &b = &a;
```

Теперь переменная **b** совпадает с **a**.

Даже начинающий программист сталкивался с ситуацией, когда в функцию необходимо передать не значение переменной, а ее адрес. Явная работа с адресами вызывает неудобства и при разработке функции, и при ее использовании. Теперь при описании функции достаточно указать, что параметр передается по ссылке, и работать с ним не как с адресом, а как с переменной.

Вот как можно описать функцию пересчета декартовых прямоугольных координат в полярные:

```
void polar (double &x, double &y, double &fi, double &r){
  r = sqrt (x*x + y*y);
  fi = atan2 (y, x);
}
```

А вот как выглядит обращение к такой функции:

```
double X, Y, Fi, R;
..........................
polar (X, Y, Fi, R);
```

Таким образом, незаметно для нас параметры передаются в функцию через их адреса.

В приведенном примере переменные X и Y тоже передаются по ссылке, хотя этого и не требует логика программы. Это сделано в целях экономии времени на их передачу, так как ссылка занимает четыре байта, а переменная типа double — восемь, и, передавая ее по ссылке, мы выигрываем вдвое (хотя в дальнейшем мы можем потерять время на обращении к этой переменной).

Самое интересное, что таким же образом можно задать и тип возвращаемого значения. Это приводит к эффекту на грани фокуса. Представьте себе, что у вас есть функция

```
double & func (int i);
```

Тогда вполне законной будет следующая запись:

```
func (10) = 1000.;
```

Этот абсурдный на первый взгляд оператор имеет вполне конкретный смысл. Такой эффект мы используем несколько позже.

На пути к классам

1. Функции — члены структуры

Как правило, описав новую структуру, вы тут же создаете набор функций, работающих с этой структурой. Например, для структуры _3d, описывающей трехмерный вектор, логично написать функцию

```
double mod(_3d & x); ,
```

которая будет вычислять модуль вектора. Это — традиционный путь языка C. Вы описываете функцию, вычисляющую модуль вектора, который передается ей через формальные параметры. Между тем модуль — характеристика, присущая каждому вектору. Поэтому

будет логично, если вектор будет как бы сам возвращать свою длину. Эту абракадабру можно проиллюстрировать следующим фрагментом программы:

```
_3d R;
double a;
............
a = mod(R); // Это традиционный подход
............
a = R.mod(); // Это новый подход
```

Здесь функция выступает как член структуры _3d. На первый взгляд, различие незначительное. Однако отсутствие видимых преимуществ не означает отсутствия здравой идеи. Здесь может получиться так же, как и в истории со структурным программированием. Не видя явных выгод, этой идеей пренебрегали, пока ошибки не привели к миллиардным потерям.

Для того чтобы функция стала членом структуры, достаточно поместить ее описание внутрь фигурных скобок, ограничивающих описание структуры. Например:

```
struct _3d {
  double x, y, z;
  double mod ();
};
```

При описании реализации функции надо после типа возвращаемого значения указать имя структуры, членом которой является данная функция, отделив от него имя функции двойным двоеточием — вот так:

```
double _3d :: mod () {return sqrt (x*x + y*y + z*z);}
```

Можно поместить реализацию функции внутрь описания структуры:

```
struct _3d {
  double x, y, z;
  double mod () {return sqrt (x*x + y*y + z*z);}
};
```

В этом случае можно опустить имя структуры, а сама функция будет считаться inline. Заметьте, что такая функция обращается с членами "своей" структуры "запросто" — по имени.

Покончив с формой, перейдем к содержанию. Наивно было бы думать, что создается новая копия функции для каждой новой переменной данного типа. Каждая функция представлена в единственном экземпляре и получает один скрытый параметр — указатель на ту переменную, для которой она вызвана (будем называть ее рабочей переменной). К этому указателю можно обратиться по имени this. Если есть оператор a = R.mod();, то this при этом вызове соответствует адресу R, а функция mod() может быть реализована так:

```
double _3d :: mod () {
  return sqrt (this->x*this->x +
        this->y*this->y +
        this->z*this->z);}
```

Если переменная не описана ни внутри функции, ни как глобальная переменная, то считается, что она является членом структуры и принадлежит рабочей переменной *this. Поэтому можно опустить указатель this и к членам структуры обращаться просто по имени.

Даже если вы не считаете нужным описывать функции как члены структур — к этому все же надо стремиться, так как такой подход создает несколько иной взгляд на программирование и формирует ваш стиль. В дальнейшем, когда речь пойдет об иерархии классов, механизме порождения и наследования, этот подход станет ключевым в понимании объектно-ориентированного программирования.

Упражнение 1:

Напишите функцию

```
double _3d :: proection (_3d r);,
```

которая будет возвращать длину проекции вектора r на рабочий вектор. Проекция вектора A на B вычисляется по формуле

```
(Ax*Bx + Ay*By + Az*Bz) / mod (B)
```

Упражнение 2:

Опишите структуру polar, определяющую вектор в полярных координатах r, fi, l. Для нее напишите функцию

```
_3d polar :: vect (); ,
```

возвращающую рабочий вектор в декартовых координатах.

Упражнение 3:

Приведите пример структуры C_Buffer, описывающей кольцевой буфер емкостью 1024 действительных числа. Для нее опишите следующие функции:

```
void init (); // Инициализация
void add (double); // Добавить элемент
double get (); // Взять элемент
int free(); // Величина свободного пространства
int used(); // Величина занятого пространства
```

Решение 1:

```
double _3d :: proection (_3d r){
  return (x*r.x + y*r.y + z*r.z) / sqrt (x*x + y*y + z*z);
};
```

Решение 2:

```
struct polar {
  double r, fi, l;
  _3d vect ();
};
_3d
polar :: vect () {
  double cf = cos(fi), sf = sin(fi), cl = cos(l), sl = sin(l);
  _3d R;
  R.x = r*sf*cl;
  R.y = r*sf*sl;
  R.z = r*cf;
  return R;
};
```

Решение 3:

Буферизация, или организация очереди, — широко распространенное техническое решение. Оно применяется, например, для согласования двух устройств или процессов, работающих в разном темпе. Один из способов реализации буфера — кольцевой буфер на одномерном ограниченном массиве.

Идея заключается в следующем. Имеется одномерный массив и две индексные переменные. Одна — индекс приемника — указывает на элемент массива, куда записывается вновь поступающее значение. При записи индекс приемника увеличивается на единицу. Вторая — индекс источника — указывает на элемент массива, из которого извлекается значение. При извлечении индекс источника также увеличивается на единицу. Если один из индексов выходит за границу массива, то он устанавливается на его начало. В организованной таким образом очереди могут "стоять" не только действительные числа, но и объекты с более сложной структурой.

В соответствии со всем вышесказанным опишем кольцевой буфер:

```
struct C_Buffer {
   double ptr [1024]; // массив буфера
   int dest, src;   // DESTination — приемник
                    // SouRCe — источник
   void init () {src = dest = 0;};
   void add (double a);
   double get ();
   int used ();
   int free ();
};

void C_Buffer :: add (double &a){
   ptr [dest++] = a;
   if (dest = = 1024) dest = 0;
};

double C_Buffer :: get (){
   if (++src ! = 1024) return ptr [src-1];
   src = 0;
   return ptr [1023];
};

int C_Buffer :: used (){
   int n = dest-src;
   if (n > = 0) return n;
   else return n + 1024;
};

int C_Buffer :: free (){
   int n = src-dest;
   if (n > 0) return n;
   else return n + 1024;
};
```

Функция init() нужна для инициализации указателей. Обращение к ней обязательно перед использованием буфера.

Пример с кольцевым буфером не является учебным. Аналогичная структура была использована автором в программах работы с модемом и с графопостроителем через последовательный порт RS-232. Предлагаемое решение вполне работоспособно и может использоваться в реальных программах.

Вместе с тем это очень характерный и показательный пример того, к чему нужно стремиться при написании программ. При реализации этой структуры удалось добиться того, что ни массив, ни указатели не используются в программе в явном виде — все необходимые операции выполняют функции.

2. Операторы overload

Раз уж мы ввели новый тип переменных — вектор, то было бы здорово иметь возможность записывать операции с векторами в виде выражений, например,

```
_3d a, b, c;
..........
a = b + c;
```

Язык С++ предоставляет такую возможность. В приведенном выражении символы '=' и '+' являются операторами, а операторы в С++ рассматриваются как функции.

Число операторов ограничено стандартным набором. Вновь вводимые операторы могут отличаться лишь типом участвующих в них операндов — отсюда и использование механизма overload. Ни приоритет, ни направление вычисления изменить нельзя. Оператор описывается так же, как и функция, только вместо имени функции пишется operator $ (где $ — описываемый оператор). Вот пример описания оператора '+':

```
_3d operator + (_3d &a, _3d &b){
  _3d c;
  c.x = a.x + b.x;
  c.y = a.y + b.y;
  c.z = a.z + b.z;
  return c;
}
```

В качестве аргументов выступают операнды, a возвращаемое значение — результат применения оператора. Бинарные операторы имеют два аргумента, причем первому соответствует левый операнд, а второму — правый. Унарные операторы имеют один аргумент.

Чтобы можно было записать выражение, с которого мы начали этот раздел, надо определить оператор '='. Это бинарный оператор; в нем участвуют два операнда, а возвратить он должен значение правого операнда (именно такую трактовку имеет этот оператор в языках С и С++).

```
   _3d& operator = (_3d &a, _3d &b) {
 a.x = b.x;
 a.y = b.y;
 a.z = b.z;
 return a;
 }
```

Здесь мы впервые возвращаем результат через ссылку. В данном случае мы можем себе это позволить, так как мы возвращаем ссылку на реальную переменную, существующую вне функции

(ссылку на которую мы получили через параметры).

Как и любая функция, оператор может быть членом структуры. Хотя это и не очевидно, но здесь тоже надо стремиться "встраивать" операторы в структуры — в дальнейшем это пригодится. Если оператор стал членом структуры, то число аргументов у него уменьшается — один из аргументов передается через переменную *this. Как уже говорилось, описание операторов отличается от описания функций лишь именем, поэтому приведем пример операторов — членов структуры:

```
struct _3d {
    double x, y, z;
    double mod();
    _3d operator + (_3d b);
    _3d & operator = (_3d b);
};
_3d _3d :: operator + (_3d b){
    _3d c;
    c.x = x + b.x;
    c.y = y + b.y;
    c.z = z + b.z;
    return c;
}
_3d & _3d :: operator = (_3d b){
    x = b.x;
    y = b.y;
    z = b.z;
    return *this;
}
```

Все сказанное про функции со полным основанием можно отнести и к операторам. Различие заключается лишь в имени: имя функции меняется на operator $ (где $ — определяемый оператор).

Скажем немного о применении ссылки на возвращаемое значение. Предположим, что мы хотим обращаться к элементам вектора по индексу. Индексу 0 будет соответствовать x, 1 — y, 2 — z. В этом случае мы можем определить следующий оператор:

```
struct _3d {
  ..............
  double &operator [] (int i){return *(&x + i);};
  ..............
};
```

После этого законными и правильными будут следующие выражения:

```
  _3d a;
  double X;
  ......
  X = a[0];
  ......
  a[0] = X;
```

Последнее выражение вызовет присваивание a.x значения X.

Приведем список операторов, которые можно переопределять:

[], (), ++, --, &, *, +, -, -, !, /, %, <<, >>,
<, >, <=, >=, ==, !=, ^, |, &&, ||, =, *=,
/=, %=, +=, -=, <<=, >>=, &=, ^=, |=.

На этом мы заканчиваем обсуждение ряда особенностей языка С++, не относящихся к его объектно-ориентированной специфике. Возможно, мы слишком долго не вводили понятия класса и объекта, зато нам удалось вынести за скобки дальнейшего изложения все детали и мелочи, затрудняющие понимание.

Упражнение 4:

Для структуры _3d определите операторы:

'-' — разность двух векторов, '^' — угол между векторами, унарный '-', '==' — равенство векторов.

Решение 4:

```
struct _3d {
    double x, y, z;
    double mod();
    _3d operator + (_3d b);
    _3d operator - (_3d b);
    _3d operator - ();
    _3d & operator = (_3d b);
    int operator == (_3d b);
    double &operator [] (int i){return *(&x + i);};
};
_3d _3d :: operator - (_3d b){
    _3d c;
    c.x = x-b.x;
    c.y = y-b.y;
    c.z = z-b.z;
    return c;
}
_3d _3d :: operator - (){
    _3d c;
    c.x = -x;
    c.y = -y;
    c.z = -z;
    return c;
}
int _3d :: operator == (_3d b){
    return (x == b.x)&&(y == b.y)&&(z == b.z);
}
```

Объектно-ориентированное программирование

1. Понятие класса

Самым интересным для нас нововведением С++ будет понятие класса. Ближайшим родственником класса является тривиальная структура. Если структуру наделить механизмом наследования, то она станет классом. Механизм наследования позволяет вновь создаваемым классам данных наследовать свойства уже существующих классов. Именно способность передавать и получать свои свойства по наследству отличает класс от структуры. Синтаксически класс описывается так же, как и структура: сначала идет ключевое слово class, затем имя класса, затем, в фигурных скобках, члены класса — данные и функции. Все, что сказано о структурах, справедливо и для классов. Прежде чем

пользоваться механизмом наследования, преобразуем уже имеющуюся у нас структуру _3d в класс:

```
class _3d {
  public :
  double x, y, z;
  double mod ();
};
double _3d :: mod () { return (sqrt(x*x + y*y + z*z)); };
```

Здесь описан класс с именем _3d. Ключевое слово public означает, что нижеследующие члены класса общедоступны. Далее описаны три действительных числа, задающих координаты вектора. Обращение к членам класса осуществляется так же, как и к членам структуры, — через точку.

Для классов применяется несколько иная терминология. Если раньше мы говорили о переменной данного типа, то теперь мы будем говорить об объекте данного класса, а функции — члены класса — будем называть методами данного класса.

2. Конструкторы и деструкторы

Создание объекта некоторого класса может быть достаточно сложной процедурой. Поэтому в языке С++ предусмотрены возможности явного описания процедур создания и уничтожения объектов данного класса. Процедуры создания объектов называются конструкторами. Процедуры уничтожения — деструкторами. Конструкторы автоматически вызываются при описании объекта, а деструкторы — при выходе из блока, в котором этот объект был описан. Конструкторы в языке С++ имеют имена, совпадающие с именем класса, и различаются между собой аргументами. Деструктор может быть только один и имеет имя, совпадающее с именем класса, которому предшествует символ '~'. И конструкторы, и деструкторы не могут иметь описания типа.

Обратимся к упражнению 3, где мы описывали кольцевой буфер. Там нам была нужна функция init для того, чтобы проинициализировать индексы источника и приемника. Эта функция обязательно должна была вызываться для каждой вновь создаваемой переменной этого типа. Если теперь мы преобразуем структуру C_Buffer в класс, то логично будет переделать функцию init в конструктор. Это пример, когда конструктор просто необходим при описании класса. Более редкий случай — когда необходимо применение деструктора. Деструктор нужен, например, для освобождения динамической памяти, занятой объектом.

Вот пример описания конструкторов класса _3d:

```
class _3d {
 _3d (double &X, double &Y, double &Z) { x = X; y = Y; z = Z;}
 _3d (_3d &a) { x = a.x; y = a.y; z = a.z; }
};
```

Если необходимые конструкторы или деструктор для класса не описаны, то транслятор создает их сам. Вызов конкретного конструктора для создаваемого объекта происходит в зависимости от аргументов, которые могут быть указаны в круглых скобках после имени создаваемого объекта. Например:

```
_3d A(0.,1.,0.),B;
```

Здесь для объекта А будет вызван описанный нами конструктор _3d (double &X, double &Y, double &Z), а для объекта В — созданный транслятором _3d ().

Существует специальный тип конструктора, который вызывается при выходе из функции, если та возвращает объект данного класса. Дело в том, что все объекты, описанные внутри функции, разрушаются (для них вызывается деструктор) при выходе из нее. Такой конструктор нужен для того, чтобы скопировать результат до того, как он будет разрушен. Это необходимо, например, для объектов, использующих динамическую память. В качестве аргумента в этом конструкторе выступает объект того же класса. Как было сказано выше, если этот конструктор не описан, то транслятор создает его сам.

Упражнение 5:

Часто требуется иметь буфер значительного объема. Поэтому нужно изменить описание класса C_Buffer так, чтобы можно было задавать объем буфера при его объявлении.

Решение 5:

В состав класса вводится дополнительное поле — длина буфера. Эта переменная подставляется везде вместо цифры 1024. Кроме того, вводится конструктор, который размещает массив ptr, и деструктор, который его освобождает.

```
class C_Buffer {
  public :
  double *ptr;  // массив буфера
  int dest,src;  // DESTination — приемник
                 // SouRCe — источник
  int len; // длина буфера
  C_Buffer (int _len = 1024);
  ~C_Buffer () {if (ptr != NULL) free (ptr);}
  void add (double a);
  double get ();
  int used ();
  int free ();
};
C_Buffer :: C_Buffer (int _len) {
  len = _len;
  dest = src = 0;
  ptr = (double *) malloc (len*sizeof(double));
  if (ptr = = NULL) len = 0;
}
void C_Buffer :: add (double &a){
  ptr [dest++] = a;
  if (dest = = len) dest = 0;
};
double C_Buffer :: get (){
  If (++src != len) return ptr [src-1];
  src = 0;
  return ptr [len-1];
};
```

```
int C_Buffer :: used (){
  int n = dest-src;
  if (n > = 0) return n;
  else return n + len;
};

int C_Buffer :: free (){
  int n = src-dest;
  if (n > 0) return n;
  else return n + len;
};
```

3. Правила доступности членов класса

При описании класса можно определять доступность членов класса для "чужих" функций. Вообще, в объектном программировании считается хорошим тоном закрывать все данные и служебные методы описываемого класса для доступа "извне".

Так, в примере с кольцевым буфером логично закрыть для доступа массив и оба указателя, чтобы с ними можно было работать только через методы add и get. Описывая класс, нужно оставлять доступ только к тем членам класса, которые необходимы для корректной работы с объектами; всю сколько-нибудь сложную работу должны брать на себя методы данного класса.

В С++ существуют три служебных слова, определяющих доступность членов класса. С первым из них вы уже встречались — это слово public. Если вы напишете это слово и поставите после него двоеточие, то все нижеследующие члены класса будут считаться общедоступными, пока не встретится другое описание доступности.

Другое слово — protected. Это слово определяет, что члены класса доступны только дружественным функциям и классам, а также классам-наследникам данного класса.

Слово private ограничивает круг "посвященных" только дружественными функциями и классами.

Дружественные функции и классы — это функции и классы, упомянутые внутри описания класса с описателем friend. Это слово ставится самым первым в описании такой функции или класса.

Упражнение 6:

Ограничьте доступ к членам класса C_Buffer так, как описано выше.

Решение 6:

```
class C_Buffer {
  protected :
  double *ptr;  // массив буфера
  int dest,src;  // DESTInation — приемник
                 // SouRCe — источник
  int len; // длина буфера
  public :
  C_Buffer (int _len = 1024);
  ~C_Buffer () {if (ptr ! = NULL) free (ptr);}
  void add (double a);
  double get ();
  int used ();
  int free ();
  int length () {return len;}
};
```

Метод length() позволяет проверить размещение массива.

4. Механизм наследования

Наследование заключается в том, что для вновь создаваемого класса мы можем указать классы, от которых он наследует их данные и методы. Такие классы мы будем называть предками, или порождающими классами, а новый класс — наследником, или порождаемым классом. Как правило, порождаемый класс имеет лишь одного предка. Иногда идеология задачи требует создания мощного дерева иерархии классов. Механизм наследования хорош отнюдь не тем, что он позволяет не описывать наследуемых членов класса. Дело в том, что транслятор выполняет скрытое преобразование типов "сверху вниз", то есть объект-наследник "сходит" за своего родителя. Иначе говоря, функции, работающие с объектами класса-предка, будут с тем же успехом работать и с объектами класса-наследника. При этом "наследники" ведут себя аналогично "предкам".

Для того чтобы задать отношения наследования между классами, надо при описании нового класса после имени класса поставить двоеточие и далее перечислить через запятую имена предков.

Предположим, что мы хотим ввести новый класс coord, описывающий систему координат в декартовом пространстве. Любая система координат задается положением центра и направлением осей. Так как центр системы координат задается вектором, то желательно, чтобы в некоторых случаях объекты класса coord вели себя аналогично объектам класса _3d. Вот пример описания класса coord:

```
class coord : public _3d {
  public :
  _3d X,Y,Z;
};
```

Здесь описан класс-наследник класса _3d. Слово public перед именем класса-предка говорит о том, что общедоступные члены предка, наследуемые порождаемым классом, также общедоступны. Членами класса coord являются действительные x, y, z координаты центра (наследуемые) и три вектора X, Y, Z, задающие направление осей в пространстве. Объект этого класса может работать и как вектор. В этом случае он представляет собой положение центра системы координат.

Упражнение 7:

Используя класс BASE_List, приведенный в приложении, опишите класс DoubleList, реализующий список из действительных чисел.

Решение 7:

Рассмотрим абстрактный класс BASE_List, приведенный в приложении. Этот класс реализует двунаправленный список. Список — это последовательность

объектов некоторого класса, называемых его элементами. В любой момент времени доступно не более одного элемента списка. Доступ к другим элементам можно получить, последовательно перемещаясь от одного элемента к другому, к концу списка или к его началу. При этом возможны три особых состояния: список пуст, доступный элемент в начале списка, доступный элемент в конце списка.

Возможна вставка нового элемента перед доступным элементом или после него. Для этого служит метод

```
ins (int l,int before); ,
```

где l — длина вставляемого элемента, before — указывает на то, что элемент должен вставляться перед доступным, если before отлично от нуля, и за ним — если равно. Вставленный элемент становится текущим. Доступный элемент можно удалить; для этого служит метод del ();. При этом становится доступным следующий к концу элемент (если его нет, то предыдущий).

Для перемещения от одного элемента к другому используются операторы: ´++´ — от начала к концу, ´--´ — от конца к началу. Метод void *object (); возвращает адрес текущего элемента.

Более подробно описание этого класса см. в приложении.

Решение заключается в том, чтобы сделать более простым доступ к текущему элементу и упростить вставку.

```
class DoubleList : virtual public BASE_List {
  public :
  int ins (int befr = 1) {
    return BASE_List :: ins(sizeof(TYPE), befr);
  }
  double& operator * () {
    return *((double *)object());
  }
};
```

5. Виртуальные методы

Как же быть, если мы хотим, чтобы "наследник" вел себя отлично от "предка", сохраняя при этом свойство совместимости с ним? На этот случай существуют виртуальные методы.

Виртуальный метод — это метод, который, будучи описан в потомках, замещает собой соответствующий метод везде — даже в методах, описанных для предка, если они вызываются для потомка.

Необходимость в применении виртуальных методов возникает, если существует набор классов, обладающих схожими по смыслу методами, различающимися лишь реализацией (методы А), и если существуют идентичные по реализации методы, использующие методы А (методы Б). В этом случае описывается базовый класс, для которого описываются виртуальные методы А. Как правило, методы А не конкретизируются (ставятся заглушки). После этого описываются методы Б, использующие методы А базового класса. Затем описываются производные классы, в которых методы А конкретизируются.

Возможен и другой взгляд на виртуальные методы. Предположим, что можно описать класс-концепцию, который послужит базовым классом. Если для класса-концепции можно указать методы, которые будут иметь различную реализацию в производных классах, то их делают виртуальными. Виртуальные методы производных классов заменяют методы базового класса везде, где они упоминаются.

Чтобы метод был описан как виртуальный, нужно перед его описанием поместить слово virtual.

Проиллюстрируем все это на примере графических объектов. Опишем класс GraphicsObject. Этот класс имеет методы Build — построить, Display — показать, Hid — скрыть и Move — переместить. Идея этого класса заключается в том, чтобы можно было перемещать графические изображения по экрану не изменяя его содержимого.

Метод Display запоминает изображение в некоторой области памяти и вызывает метод Build, который в этой области строит новое изображение. Метод Hid скрывает изображение, построенное методом Build, восстанавливая то изображение, которое запомнил метод Display. Наконец, метод Move вызывает метод Hid, чтобы скрыть старое изображение, и метод Display, чтобы построить его на новом месте.

Объявим метод Build виртуальным. Опишем теперь двух потомков класса GraphicsObject. Первый — Circle — имеет метод Build, который строит кружок. Второй — Rectangle — имеет метод Build, который строит прямоугольник. Все! Если теперь мы объявим

```
  Circle A;
  Rectangle B;
```

то вызывая методы Display, Hid и Move для объекта A, мы будем работать с кружком, а для объекта B — с прямоугольником.

```
// Описание класса GraphicsObject
class GraphicsObject {
  protected:
  int _x, _y;
  void *image;
  public:
  void Display (int x, int y);
  virtual void Build (int x, int y);
  void Hid ();
  void Move (int x, int y);
};

// Описание класса Circle
class Circle : virtual public GraphicsObject{
  virtual void Build (int x, int y);
            // метод строит кружок
};

// Описание класса Rectangle
class Rectangle : virtual public GraphicsObject{
  virtual void Build (int x, int y);
            // метод строит прямоугольник
};
```

Если бы метод Build не был объявлен виртуальным, то при вызове A.Display вызывался бы метод

```
GraphicsObject :: Build (int,int);
```

Но поскольку Build — виртуальный, то вызывается

```
Circle :: Build (int,int);
```

Как же реализуется механизм виртуальных методов? Каждый объект, помимо полей данных, описанных для данного класса, содержит ссылку на таблицу адресов виртуальных методов своего класса. При вызове виртуального метода его адрес извлекается из соответствующей данному объекту таблицы — таким образом вызывается "то, что надо". Объекты "таскают" свои виртуальные методы с собой.

Вот и все. Конечно, изложенный материал плохо согласуется со сложившейся практикой программирования. Объектно-ориентированное программирование станет, возможно, самым ценным приобретением алгоритмических языков за всю их предыдущую историю. Новые идеи требуют слишком радикального изменения взглядов и подходов к программированию. Применять объектное программирование нужно начинать постепенно, исподволь, там, где этого вроде и не требуется — это позволит во всеоружии подойти к более серьезным задачам. Не нужно бояться экспериментировать и применять новое. Помните, что смелость не только города берет, но и помогает осваивать C++.

А.Матвеев

Компания Convex Computer Corporation выпустила то, что она называет первым коммерческим визуальным отладчиком, способным анализировать сильно оптимизированный код программы на уровне языков высокого уровня.

Новый продукт, названный CXdb, может работать с программами на FORTRAN'е и C с использованием оконного интерфейса на графических терминалах в среде системы CXWindows. Пользователь может в разных окнах одновременно видеть объектный код, исходный текст и результат работы программы. Он также получает доступ ко всем системным ресурсам.

CXdb продается с начала июля. Минимальная цена — 9500 долларов. Convex производит суперкомпьютеры, предназначенные в основном для технических и научных нужд.

Newsbytes News Network,
19 July, 1991

Президент фирмы Borland Филипп Кан высказал свою точку зрения на продолжающиеся судебные процессы между Borland и Lotus, а также между Ashton-Tate и Fox Software на собрании группы пользователей IBM в Пасадене, Калифорния.

По слова Кана, оба дела тесно связаны между собой. В обоих случая истец, Lotus и Ashton-Tate, пытаются защитить авторским правом не новое изобретение, а иерархическую структуру команд, определенную самим функционированием программы.

Он добавил, что компании могли бы получить патент на свои творения, но они стремятся заполучить исключительные права на то, что, по мнению президента фирмы, не может защищаться никаким авторским правом.

Если истцы выиграют процессы, это может изменить не только компьютерную индустрию, но и весь товарный рынок. Внезапно производитель самолетов заявит, что он обладает авторскими правами на размещение органов управления в пилотской кабине, чем заставит других производителей располагать их по-другому. Видеомагнитофоны, телевизоры, все имеют иерархию команд, базирующуюся на выполняемых ими функциях. Если эти команды станут объектом авторского права, то автоматически каждый новый производитель будет вынужден выдумывать свои команды, делая пользование устройствами нескольких фирм почти невозможным.

"Если иерархия команд, основанная на функциональных свойствах будет защищаться авторским правом, это будет катастрофой",- сказал Филипп Кан.

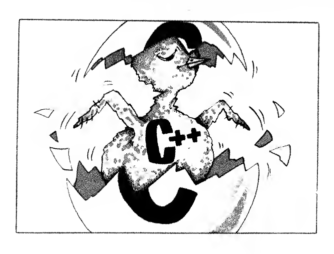
Он добавил, что Borland мог бы платить фирме Lotus небольшие отчисления за использование популярного командного интерфейса, но Lotus хочет добиться судебного решения о том, что они являются собственниками этих команд и никто кроме них не может их использовать.

"Мы, конечно, можем просто убрать этот дополнительный элемент из Quattro Pro, и большинство наших пользователей от этого не пострадает. Но теперь для нас этот процесс стал просто принципиальным".

Комментируя процесс между Ashton-Tate и Fox Software, Кан ехидно отозвался об адвокатах обеих сторон и сказал, что они не скандальная компания и с Fox договорятся быстро.

Ранее в июле Borland объявил о намерении купить фирму Ashton-Tate. А затем и сделал это.

Newsbytes News Network,
23 July, 1991



"Науки, связанные с вычислительной техникой, стареют, и идеи, которые развивались в 60-е и 70-е годы, сегодня являются лишь элементарной основой идей и методов."

Дж.Ульман "Computational aspects of VLSI", Computer Sciense Press, 1987.

СОВРЕМЕННЫЕ МЕТОДЫ ПРОМЫШЛЕННОЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Чтобы немного прояснить разницу между современностью и недалеким прошлым, я позволю себе перечислить некоторые основные достижения в науке о программном обеспечении компьютеров, которые сейчас позволяют нам работать более производительно и более уверенно смотреть в будущее.

К концу 70-х годов был достигнут ряд важных результатов:

- создание формальной теории проектирования компиляторов с алгоритмических языков;
- формирование концепции абстрактных типов данных;
- разделение программного обеспечения на аппаратно-зависимую и проблемно-зависимую части;
- создание переносимой операционной системы UNIX и переносимого языка C;
- разработка унифицированных runtime-интерфейсов для прикладных программ (API) и стандартных форматов обмена данными.

Между тем, растущее применение компьютеров (в особенности персональных) в самых различных областях приводило к росту потребности в разработке программного обеспечения.

В 70-е годы стало ясно, что прогресс в области системного программирования должен быть распространен и на методы разработки прикладного ПО. Таким распространением явилось внедрение в практику возникшей на основе концепции абстрактных типов данных технологии объектно-ориентированного программирования.

Объектно ориентированное программирование — не новый термин. Мы слышим о системах ООП уже с конца 70-х годов. Но подлинное десятилетие ООП началось только сейчас, с созданием и широким внедрением эффективных объектно-ориентированных языков, таких как C++ и объектно-ориентированных версий Pascal.

Объектно-ориентированные языки 70-х (например, такие как Smalltalk) опирались на мощные и громоздкие интегрированные среды и реализовывались в значительной мере на принципах интерпретации. На сегодняшний же день основным языком ООП стал C++, созданный группой под руководством Бьерна Страуструпа из Bell Laboratories (AT&T, USA). Значение этой разработки нужно рассматривать исходя из того, что впервые была предложена стандартная, базирующаяся на синтаксисе переносимого языка, форма для кодирования объектов и их методов. Более того, идеология C++ была нацелена на то, чтобы все операции по связыванию объектов и манипулирующих ими методов выполнялись во время компиляции. В результате того, что в C++ механизм скрытия данных работает во вре-

мя компиляции, получающаяся исполняемая программа (.exe файл) не содержит ничего лишнего (например, "монитора виртуальных объектов" или "динамического интерпретатора методов класса").

Продолжающееся успешное внедрение C++ технологии в коммерческие программы позволяет с уверенностью утверждать, что язык C++ стал основным средством разработки программных систем, ориентированных на конечного пользователя.

Системами, ориентированными на конечного пользователя, автор называет программные системы, которые могут быть быстро освоены персоналом с минимальными навыками работы с компьютером (это такие системы, как dBASE, Clipper, ACAD, Framework и т.д.). Язык C++ является средством разработки подобных систем.

Некоторые уже достаточно хорошо известные зарубежные программные разработки имеют вполне отчетливый "привкус" C++. Еще одним примером успешного применения методов ООП в индустрии программного обеспечения может служить создание в недавнее время целого ряда специализированных генераторов исходных текстов программ или их частей (это реализуется путем компоновки объектов из библиотеки).

Возьмем, например, построитель интерфейса в системе ПО компьютера NeXT. Эта программа позволяет описывать пользовательский интерфейс прикладной программы, оперируя понятиями вида: окно, функциональная клавиша, меню и т.д. Построитель интерфейса NeXT генерирует на выходе исходный текст на диалекте C++. Другим примером может служить возможно уже знакомый читателю построитель интерфейса из пакета Turbo Pascal 6.0. Джефф Уолден[11], описывая пакет Toolbook фирмы Asymetrix, который "позволяет строить свою программу, компонуя стандартные объекты", говорит о целом ряде аналогичных продуктов конкурирующих фирм.

Хочу обратить внимание читателя на то, что время подобных построителей (к слову сказать, существенно облегчающих жизнь их покупателям) пришло с внедрением ООП. Овладев одним из современных языков ООП (например, Zortech C++, Borland C++ 2.0 или Turbo Pascal 5.5), вы сами почувствуете, с какой легкостью могут быть написаны подобные программы. Время "построителей" (причем не только интерфейсов пользователя) пришло с внедрением C++. Чтобы в какой-то мере проиллюстрировать технологию программирования с использованием объектно-ориентированного подхода, предлагаем вам пример разработки небольшой программы на C++.

Допустим, мы хотим разработать программу, реализующую простейшие функции меню интерфейса. Что нам нужно для этой программы? Очевидно, необходим объект меню. Какие действия можно выполнять с меню? Например, следующие:
- создать пустое меню;
- добавить строку к меню;
- нарисовать меню на экране дисплея;
- производить выбор пунктов меню (точнее, передать управление меню пользователю за клавиатурой);
- удалить меню с экрана.

Преимущество объектно-ориентированного подхода состоит в том, что он позволяет нам рассматривать такой сложный объект, как меню, в качестве элементарного типа данных. С этим объектом мы можем оперировать так же просто, как с обычной переменной, описанной, например, так:

```
int j ;
```

В принципе, меню — такой же простой объект, как и любая переменная. Сущность меню определяется действиями, которые можно над ним выполнять. Действия над объектом в C++ называются методами. Нам необходимо определить эти методы формально, т.е. запрограммировать их.

Для того чтобы запрограммировать все перечисленные выше методы, нам понадобятся некоторые данные, отражающие содержимое и текущее состояние меню. Эти данные включают в себя следующие элементы:
- массив строк меню;
- текущее число строк в меню;
- номер текущей (подсвеченной) строки меню;
- координаты левого верхнего угла меню на экране;
- код нажатой пользователем клавиши.

Язык C++ позволяет нам рассматривать данные объекта и методы объекта совместно. Закодировав данные, и запрограммировав методы, мы определим не просто отдельный объект меню, а любое произвольное меню. Мы зададим целый класс объектов. Совокупность данных и оперирующих ими методов в C++ оформляется специальным образом синтаксически и называется описанием класса.

Составим описание класса меню в синтаксической нотации C++.

```
class menu_vert
{
private:
   int x ;              // Текущий левый верхний угол меню
   int y ;
   int present_item ;      // Номер текущей строки меню
   int number_of_item ;    // Текущее число строк меню
   int x_width ;         // Ширина меню
   int glob_menu_key ;     // Выбирающая клавиша
   int glob_menu_item ;    // Номер выбранной строки меню
   char item[MAX_ITEM_AMOUNT][MAX_ITEM_WIDTH] ; // Строки меню
public:
void init()  ;                // Инициализировать пустое меню
void add_item( char *newitem ) ;  // Добавить строку в меню
void draw_menu()  ;              // Нарисовать меню на экране
void menu_choice()  ;            // Выбирать меню
void erase_menu() ;             // Стереть меню с экрана
void set_menu_xy( int inx , int iny ) ;  // Установить положение
                                // меню на экране
int get_glob_menu_item() ;       // Получить номер выбранной строки меню
int get_glob_menu_key() ;        // Получить код клавиши, которой
                          // производился выбор (например, Esc или Enter)
} ;
```

Все, что описано выше слова public является скрытым в меню и недоступным для любого участка программы. Доступны лишь методы класса, описанные ниже public. Полностью программа приведена в приложении к статье. Там вы найдете и реализацию методов класса menu_vert. А здесь мы покажем, как легко

теперь использовать в программе самые разнообразные меню.

Допустим, мы хотим создать меню оболочки компилятора. Нет ничего проще, ведь у нас уже есть класс меню.

```
main()
{
  .
  .
 menu_vert   abcd1 , abcd2 , abcd3  ;        // Опишем три меню
         // Не правда ли, не сложнее чем  "int i, j, k;"?

  abcd1.init() ;                         // Инициализируем первое меню
  abcd1.add_item("ЗАГРУЗИТЬ") ;
  abcd1.add_item("НОВЫЙ ФАЙЛ") ;
  abcd1.add_item("СОХРАНИТЬ") ;
  abcd1.add_item("ЗАПИСАТЬ В ...") ;
  abcd1.add_item("КАТАЛОГ") ;
  abcd1.add_item("ВЫХОД В DOS") ;
  abcd1.add_item("ПОКИНУТЬ - Alt X") ;
  abcd1.set_menu_xy(1,1) ;

  abcd2.init() ;                         // Инициализируем второе меню
  abcd2.add_item("КОНТРОЛЬ СКОБОК") ;
  abcd2.add_item("КОНТРОЛЬ СИНТАКСИСА") ;
  abcd2.add_item("ВЫПОЛНИТЬ ПРОГРАММУ") ;
  abcd2.set_menu_xy(20,1) ;

  abcd3.init() ;                         // Инициализируем третье меню
  abcd3.add_item("СПИСОК СИНТАКСИЧЕСКИХ ОШИБОК") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ ЛИТЕРАЛЫ") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ КОНСТАНТЫ") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ СТАНДАРТНЫЕ ФУНКЦИИ") ;
  abcd3.add_item("ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ") ;
  abcd3.set_menu_xy(43,1) ;
  .
  .
// Теперь, если пользователь должен вступить в диалог с меню 2
  abcd2.draw_menu() ;      // Нарисуем меню,
  abcd2.menu_choice() ;    // затем передадим управление пользователю
                   // Так же просто, как  "i=i+1 ;"

   user_string_number = get_glob_menu_key() ;  // Получаем результат
   user_key = get_glob_menu_item() ;          // диалога с пользователем

   abcd2.erase_menu() ;                    // Удаляем меню
   if ( user_string_nubmer  .......  и т.д.
}
```

Вся утомительная возня с массивами и счетчиками по модулю N скрыта внутри методов класса, причем она автоматически воспроизводится, стоит лишь описать новую переменную меню.

Класс в C++ наиболее близок к шаблону структуры в языке С. Переменные структурного типа (как abcd1, abcd2, abcd3) называются экземплярами класса. Отличие классов в C++ от структур в С заключается в том, что в C++ область видимости переменных, принадлежащих структуре-экземпляру класса, находится исключительно внутри методов класса. Т.е. идентификаторы present_item, x_width и т.д. имеют смысл только внутри функций: init(),..., get_glob_menu_key(). Причем вызов abcd1.init() оперирует с abcd1.present_item, а вызов abcd2.init() с abcd2.present_item и т.п.

Вызов abcd2.init() означает, что функция menu_vert::init() получает в качестве параметра указатель на abcd2, — экземпляр класса menu_vert.

Так работает механизм скрытия данных C++. Скрытие данных позволяет избежать многих видов ошибок, сохраняет концептуальную целостность объектов, проясняет структуру больших программ.

Умело применяя механизм скрытия, можно писать довольно сложные программы, оперирующие сложноструктурированными данными. Причем на C++ такие программы можно писать достаточно быстро. C++ значительно повышает повторную используемость разработанных алгоритмов. Этим целям, в частности, служит механизм наследования классов и виртуальных функций. Возможности хорошо написанной объектно-ориентированной программы на C++ могут быть существенно расширены или изменены, с модификациями исходного текста на порядок меньшими, чем, например, при использовании обычного (не ОО) языка программирования. C++ позволяет писать очень гибкие программы. В связи с этим давайте вернемся к нашему примеру с меню.

Представьте себе, что вам теперь необходимо иметь меню не из текстовых строк, а из графических изображений на экране дисплея. Данные и методы класса для графического меню почти так же просты, как и для текстового. Например, этот класс может быть таким:

```
class slide_menu
{
private:
 char slb_file_name[FILENAME_LENGTH] ;  // Имя файла, содержащего
                              // библиотеку изображений пунктов меню
 int max_item ;                     // Реальное текущее число
                              // установленных пунктоа меню
 int present_item ;                 // Текущий пункт для menu_choice
 sld_menu_item  *items[MAX_ITEMS_AMOUNT] ;  // Изображения меню
 int ret_codes[MAX_ITEMS_AMOUNT] ;        // Коды, возвращаемые
                                   // вызывающей программе
public:
 slide_menu( char *slb_f_n ) ;           // Создать пустое меню
 void set_sld_item( sld_menu_item  *x , int ret_c ) ;  // Добавить
                                        // изображение в меню
 int fill_menu( input_buffer &buffered_slb ) ; // Заполнить адреса поиска
                                   // (найти адреса всех изображений
                                   // меню в оглавлении библиотеки)
 void draw_menu ( input_buffer &buffered_slb );  // Нарисовать меню
 int menu_choice(int &key_code ) ;                // Выбирать меню
                              // (возвращает текущий выбранный пункт
                              // и код выбирающей клавиши)
 void menu_erase() ;                   // Удалить меню
} ;
```

Назначение методов аналогично текстовому меню: создать пустое меню, добавить пункт, нарисовать меню, выбирать пункт меню. Отличие в том, что сам пункт графического меню представляет собой более сложную структуру — экземпляр класса sld_menu_item. Спецификация этого класса приведена ниже:

```
class sld_menu_item
{
private:
 char member_slb_name[MAXCHNUM] ;  // Имя изображения в каталоге
                          // библиотеки
 long seek_addr_in_slb ;    // Стартовый адрес изображения в библиотеке

 int  left_up_corner_X ;    // Границы  области экрана, в которой
 int  left_up_corner_Y ;    // будет нарисовано изображение
 int  right_down_corner_X ;
 int  right_down_corner_Y ;

 int  deltX_slide_file ;    // Экранное соотношение изображения
 int  deltY_slide_file ;
public:
   sld_menu_item(char *m_s_name,int l_u_X,int l_u_Y,
      int r_d_X,int r_d_Y) ;         // создать пункт графического меню
   void direct_fill_seek_addr ( long s_addr ) ;
                              // Установить адрес начала графических
                              // данных пункта меню в файле
библиотеки
   void draw_menu_item( int buffered_slb ) ;  // Нарисовать пункт меню
   void menu_item_activate() ;             // Выделить пункт меню
                              // (подсветить или взять в рамку)
   void menu_item_deactivate() ;           // Убрать подсветку или рамку

   void erase_menu_item() ;                // Удалить пункт меню с экрана
   void set_menu_item_borders( int lux , int luy ,
       int rdx , int rdy ) ;  // Установить пределы области изображения
                      // пункта меню на экране
} ;
```

Обратите внимание на то, что при разработке методов класса slide_menu нам необходимо знать только спецификацию вызова методов класса (т.е. только секцию public спецификации класса). Теперь для прорисовки пункта в методе draw_menu класса slide_menu нам необходимо вместо

```
gotoxy( «X i-го пункта меню» , «Y i-го пункта меню» ) ;
cputs( «строка i-го пункта меню» ) ; ,
```

как в случае текстового меню, писать

```
(*items[i]).draw_menu_item( lib_handle ) ; ,
```

где lib_handle — указатель на открытый файл библиотеки, откуда нужно читать графические данные.

Конкретный вид программы вывода графического меню (метод draw_menu класса slide_menu) может быть, например, таким:

```
void slide_menu::draw_menu( int buffered_slb )
{
// Вычислить все текущие углы слайдов по процедуре
// item_coordinates, затем отрисовать все слайды меню

 int ic ;                    // счетчик пунктов
 int lux , luy , rdx , rdy ;

 for ( ic=0 ; ic < max_item ; ic++ )
 {
 // Вычисление координат углов всех пунктов (и их установка)
 item_coordinates( max_item , ic , &lux , &luy , &rdx , &rdy ) ;
 (*items[ic]).set_menu_item_borders (lux,luy,rdx,rdy) ;
 } ;

 clearviewport() ;   // порт вывода и input_buffer устанавливаются для
                     // меню или меню системы в вызывающей программе

 for ( ic=0 ; ic < max_item ; ic++ )
  {
   // отрисовать все пункты
   (*items[ic]).draw_menu_item( buffered_slb ) ;
  } ;
}
```

Метод draw_menu_item класса sld_menu_item выводит графический пункт меню на экран путем чтения и интерпретации структуры графического файла, выполняя при этом масштабные преобразования. Это довольно сложная программа, но пользоваться ей просто.

Таким образом, мы подходим к еще одной замечательной особенности объектно-ориентированных языков. ОО языки позволяют разбить прикладную задачу на уровни программного обеспечения (подобно принципу уровневого построения эталонной модели программного обеспечения взаимодействия открытых систем). Каждый уровень имеет четко определенный лаконичный интерфейс и набор услуг (методов), предоставляемых верхнему уровню. Используя механизм виртуальных функций C++ можно организовать интерфейс между уровнями так, что исходные тексты верхнего уровня не будут зависеть от структуры объектов нижнего уровня, с которыми они оперируют. В этом случае одна и та же программа позволяет оперировать с существенно различными данными. Пример из области организации интерфейса пользователя достаточно нагляден. Но тот же подход распространим и на любые задачи. Так, например, программа расчета по методу МКЭ может не зависеть от вида самих конечных элементов; сетевая операционная система может работать без изменения с различными протоколами нижнего уровня и т.д.

Перед нами открывается перспектива формализации алгоритмических знаний человечества в различных прикладных областях на основе единых инженерных средств. Мы станем свидетелями создания огромных библиотек прикладных объектов с четко определенными интерфейсами, созданных в переносимой, не зависящей от оборудования форме.

Разбиение на уровни и спецификация уровней в прикладном программном обеспечении скорее всего станет следующим объектом стандартизации в компьютерной науке.

Особенностью современного этапа компьютеризации в нашей стране является то, что теперь с компьютером все чаще вынуждены взаимодействовать люди неподготовленные — бухгалтеры, конструкторы, технологи, рабочие и многие другие. Эти люди несут громадный обьем прикладных знаний. И применение ими новых методов компьютерной обработки информации должно дать огромный эффект.

Неподготовленный пользователь должен работать с компьютером, оперируя привычными для него понятиями и образами. Этот принцип используется в разработанном авторами пакете AutoMENU, который предоставляет программисту развитые средства графического взаимодействия с пользователем.

Опыт показывает, что применение объектно-ориентированных методов в разработке программного обеспечения дает следующие результаты:

- повышается индивидуальная производительность труда программиста;
- возрастают возможности проектирования (используя C++, программист вполне может разрабатывать программу объемом 25 000 строк и более).
- увеличивается время начального этапа проектирования (продумывание структуры и функций программы);
- существенно уменьшается время отладки программы;
- на порядок уменьшается время разработки новых функций и внесения изменений в программы. Чем тщательнее проведено предварительное проектирование, тем больше возможностей по внесению изменений в готовую разработку;
- ОО программа лучше приспособлена для ее разработки коллективом программистов;
- существенно повышается надежность функционирования программ. Логические ошибки практически полностью устраняются на стадии предварительного проектирования;
- возрастают возможности по использованию в новых разработках частей работающих программ (с их минимальными модификациями).

Важным и продуктивным направлением исследований может стать создание проблемно-ориентированных систем автоматической генерации программ из библиотек типовых модулей, на основе объединения концепций C++ и возможностей экспертных систем.

Б.Ткаченко

По материалам:

Stanley, Lippman "C++ Primer", AT&T Bell Laboratories, Addison Wesley Publishing company, 1989
Richard S. Weiner and Lewis Pinson, "An Introduction to Object-Oriented Programming and C++", MA, Adisson Wesley, 1988
Stanley, Lippman, Stroustrup "Pointers to Class Members in C++", USENIX C++ Conference, October, 1988.
Bjarne Stroustrup "The C++ Programming Language", Addison Wesley, Reading, MA, 1986.
Adle Golberg and David Robson "SMALLTALK-80: The language and its implementation", Addison-Wesley, MA, 1983.
Ralph Johnson and Brian Foote "Designing Reusable Classes", Journal of Object-Oriented Programming, June/July 1988.
Turbo C++ 1.0 Programmer's Guide, Borland International, CA, USA.

```
// Приведенная ниже программа тестировалась в среде Turbo C++ 1.0.
// Методы, декларируемые в описании класса ниже функции windpop,
// в данном примере не используются, но они иллюстрируют то,
// как, добавляя новые методы, можно сделать интерфейс меню
// более гибким.
//      Copyright B.Tkacheako,1990,Dniepropetrovsk city

#define  MAX_VAR_NUMBER     160

#define  COL_ORDINARY       GREEN
#define  BK_ORDINARY        BLACK

#define  COL_SELECTION      WHITE
#define  BK_SELECTION       MAGENTA

#define  COL_ALARM          WHITE
#define  BK_ALARM           RED

#define  COL_ALARM_SEL      BLACK
#define  BK_ALARM_SEL       LIGHTGRAY

#include <mem.h>
#include <dos.h>
#include <string.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
      // для exit
#include <math.h>

#include "keyboard.h"

// Константы класса MENU_VERT

#define MAX_ITEM_AMOUNT        20
#define MAX_ITEM_WIDTH         80
#define BORDER_COLOR           COL_ORDINARY
#define FRAME_COLOR            COL_ORDINARY
#define SELECTION_COLOR        COL_SELECTION
#define BORDER_BK_COLOR        BK_ORDINARY
#define FRAME_BK_COLOR         BK_ORDINARY
#define SELECTION_BK_COLOR     BK_SELECTION

// Определение класса MENU_VERT
class menu_vert
{
private:
   int x ;  // Текущий левый верхний угол меню
   int y ;
   int present_item ;      // Текущий (подсвеченный) пункт меню
   int number_of_item ;      // Текущее число пунктов меню
   int x_width ;          // = max ( strlea ( item all ))
   int glob_menu_key ;       // Выбирающая клавиша
   int glob_menu_item ;      // Выбранный пункт меню
   char item[MAX_ITEM_AMOUNT][MAX_ITEM_WIDTH] ; // Пункты меню
   char save_area[(MAX_ITEM_AMOUNT*MAX_ITEM_WIDTH)*2+10] ;
                                         // Область сохранении экрана
public:

void init() ;                     // Инициализировать пустое меню
void add_item( char *newitem ) ;       // Добавить строку в меню
void draw_menu() ;              // Нарисовать меню с сохр. экрана (под ним).
void menu_choice() ;            // Выбирать пункты меню
void erase_menu() ;             // Удалить меню с восстановление экрана
void set_menu_xy( int inx, int iny ) ;  // Установить левый верхний угол
int get_glob_menu_key() ;         // Получить клавишу выбора строки меню
int get_glob_menu_item() ;         // Получить номер выбранной строки

// Служебные функции сохранении старого экрана
void windsw( int x1 , int y1 , int x2 , int y2 ) ;
void windpop( int x1 , int y1  , int x2  , int y2 ) ;

int get_x_width() ;          // Функции доступа к скрытым данным класса
int get_number_of_item() ;
void set_item( int item_number , char *newitem ) ;   // Изменить
                                        // существующий пункт
void set_present_item ( int item1 ) ;   // Принудительно установить номер
                                  // текущего пункта меню
void hide_items_cursor() ;    // Убрать подсветку текущего пункта
void draw_items_cursor() ;     // Снова подсветить текущий пункт
void clear_menu() ;           // Прописать все пункты пробелами
) ;

// Функции используются членами класса MENU_VERT
void draw_box(
        int    typ   ,
                          /*        0- double line */
                          /*  3 - space characters only */
        int    col   ,   /*  color of the box border */
        int    colbk   , /*  color of the box border */
        int    y1  ,     /* x1 < x2  ; y1 < y2  */
        int    x1  ,
        int    y2  ,
        int    x2
               )
{
 int i  ;
 int dx , dy  ;
 char   *right_upper , *hor_lin , *left_upper , *vert_lin ,
          *right_bottom , *left_bottom ;
switeh ( typ )
{
case 0: {
 left_upper   ="х";
 right_upper  ="ы";
 vert_lin     ="ъ";
 hor_lin      ="щ";
 right_bottom ="ь";
 left_bottom  ="ф";
      } ; break ;
case 3: {
 left_upper   =" ";
 right_upper  =" ";
 vert_lin     =" ";
 hor_lin      =" ";
 right_bottom =" ";
 left_bottom  =" ";
      ) ; break ;
} ;
textcolor(col) ;
textbackground(colbk) ;

dx = x2 - x1 ;
dy = y2 - y1 ;
gotoxy(x1,y1) ;
cputs(left_upper) ;
for ( i=1 ; i<= dx-1 ; i++ ) cputs(hor_lin) ;
cputs(right_upper) ;
for ( i=1 ; i<= dy-1 ; i++ )
  {
  gotoxy(x2,y1+i) ;
  cputs(vert_lin) ;
  gotoxy(x1,y1+i) ;
  cputs(vert_lin) ;
  ) ;
gotoxy(x1,y2) ;
cputs(left_bottom) ;
for ( i=1 ; i<= dx-1 ; i++ ) cputs(hor_lin) ;
cputs(right_bottom) ;
}

int inc_pos(int *cur_x , int *cur_y , int *x1 , int *x2 , int *y2 )
/* Сервисная функция для процедур WINDSW and WINDPOP */
(
 int ine_pos_var ;
 inc_pos_var = 1 ;
 (*eur_x)++ ;
 if ( (*cur_x) > (*x2) )
  {
   (*cur_y)++ ;
   (*cur_x) = (*x1) - 1 ;
  ) ;
 if ( (*cur_y) > (*y2) ) inc_pos_var = 0 ;
return( inc_pos_var ) ;
}

/* Процедура временного сохранения окна */
void menu_vert::windsw( int x1 ,
    int y1 ,
    int x2 ,
    int y2
    )
#define  ScreenBuffer  0xb800      /* $b000 — монохромный режим */
{
int  i, cur_x , eur_y  ;
    cur_x = x1 -2 ;
```

```
    cur_y = y1 -1 ;
    i = 0 ;
while ( inc_pos( &cur_x , &cur_y , &x1 , &x2 , &y2 ) )
  {
movedata(ScreenBuffer,(cur_x+cur_y*80)*2,FP_SEG(&save_area[i]),
                         FP_OFF(&save_area[i]),2) ;
  i = i + 2 ;
  };
  return ;
}

/* Процедура восстановления окна */
void menu_vert::windpop(
          int     x1 ,
          int     y1 ,
          int     x2 ,
          int     y2
        )
#define  ScreenBuffer  0xb800    /* $b000 — монохромный режим */
{
int  i, cur_x , cur_y  ;
   cur_x = x1 -2 ;
   cur_y = y1 -1 ;
   i = 0 ;
   while ( inc_pos( &cur_x , &cur_y , &x1 , &x2 , &y2 ) )
   {
movedata(FP_SEG(&save_area[i]),FP_OFF(&save_area[i])
                 ,ScreenBuffer,(cur_x+cur_y*80)*2,2) ;
     i = i + 2 ;
   };
}

// Методы класса MENU_VERT
int menu_vert::get_x_width ()
   {
  return(x_width)  ;
   }

int menu_vert::get_number_of_item()
   {
  return(number_of_item)  ;
   }

int menu_vert::get_glob_menu_key()
   {
  return(glob_menu_key) ;
   }

int menu_vert::get_glob_menu_item()
   {
  return(glob_menu_item) ;
   }

void menu_vert::init()
// инициализация вертикального меню
{
     x = 0 ;
     y = 0 ;
     number_of_item  = 0 ;
     present_item  = 0 ;
     x_width  = 0 ;
     glob_menu_key  = 0x20 ;
     glob_menu_item  = 0 ;
}

void menu_vert::clear_menu()
{
  int i , j  ;
    for ( i=1 ; i<=number_of_item ; i++ )
    {
     item[i][0] = 0x00 ;
     for ( j=1 ; j<= x_width ;  j++ )  strcat(item[i]," ") ;
    };
}

void  menu_vert::menu_choice()
// выбор в вертикальном меню
{
 int      i ;
 int      j  ;
 int      user_key   ;
 int      previous_item   ;

     user_key  = 0x20 ;

     while  (
           ( user_key != EASC_Enter )      &&
           ( user_key != EASC_Left )      &&
           ( user_key != EASC_Right )      &&
           ( user_key != EASC_Esc )
           )
{
     user_key  = i_readkey() ;
     previous_item  = present_item ;

   switch ( user_key )
    {
    case      EASC_Home    :
                    {
                    present_item  = 0 ;
                    } ; break ;
    case      EASC_Left    :
                    {
                    } ; break ;
    case      EASC_End     :
                    {
                    present_item  = number_of_item - 1 ;
                    } ; break ;
    case      EASC_Up      :
                    {
                    if ( present_item  == 0 )
                    present_item  = number_of_item ;
                    if ( present_item  > 0 )
                    present_item  = present_item - 1 ;
                    } ; break ;
    case      EASC_Down    :
                    {
                    present_item  = present_item + 1 ;
                    if ( present_item  >= number_of_item )
                    present_item=0 ;
                    } ; break ;
    case      EASC_PgUp    :
                    {
                    } ; break ;
    case      EASC_Right  :
                    {
                    } ; break ;
    case      EASC_PgDn    :
                    {
                    } ; break ;
   } ;  // switch

           textcolor(FRAME_COLOR) ;
           gotoxy(x+1,y+1+previous_item) ;
           cputs(item[previous_item]) ;
           textcolor(SELECTION_COLOR) ;
           gotoxy(x+1,y+1+present_item) ;
           cputs(item[present_item]) ;

     glob_menu_key = user_key ;
     glob_menu_item = present_item ;

}  ;      // while
}                 // метод menu_choice

void menu_vert::draw_menu()    // с сохранением предыдущего экрана
{
 int      i , j , y2 , x2 ;

     y2= y + number_of_item  + 1 ;
     x2= x + x_width + 1 ;
     (*this).windsw(
               x,
               y,
               x2,
               y2
          ) ;

     gotoxy(x,y) ;
     textcolor(BORDER_COLOR) ;
     textbackground(BORDER_BK_COLOR) ;

     draw_box( 0,
             BORDER_COLOR,
             BORDER_BK_COLOR,
             y,
             x,
             y2,
             x2
          ) ;

     textcolor(FRAME_COLOR) ;
     textbackground(FRAME_BK_COLOR) ;
      for ( i=0 ; i<=number_of_item - 1 ; i++ )
        {
          if ( i == present_item )
          {     textcolor(FRAME_COLOR) ;
                textbackground(FRAME_BK_COLOR) ;
          } ;

          gotoxy(x+1,y+1+i) ;

          for ( j=x+1 ; j<=x+x_width ; j++ )
          {
            gotoxy(j,y+1+i) ;
            cputs(" ") ;
          } ;
          gotoxy(x+1,y+1+i) ;
          cputs( item[i] ) ;
          textcolor(FRAME_COLOR) ;
          textbackground(FRAME_BK_COLOR) ;
        } ;
} .

void menu_vert::set_menu_xy( int inx , int iny )
  // установить левый верхний угол
{
     x = inx ;
     y = iny ;
}

void menu_vert::add_item( char *newitem )
{
     strcpy(item[number_of_item],newitem) ;
     if ( x_width < strlen(newitem) ) x_width = strlen(newitem) ;
     number_of_item = number_of_item + 1 ;
```

```
}

void menu_vert::set_item( int item_number , char *newitem )
// Изменить значение существующей строки меню
{
 if ( item_number < number_of_item )
 { strcpy(item[item_number],newitem) ;
 }
else
 {
 strcpy(item[number_of_item],newitem) ;
 } ;
}

void menu_vert::erase_menu()
// без восстановления предыдущего экрана
{
 int    i , j , y2 , x2 , y1 , x1 ;

     y2 = y+number_of_item+1 ;
     x2 = x+x_width+1 ;
     y1 = y ;
     x1 = x ;

    (*this).windpop(
              x1,
              y1,
              x2,
              y2
         ) ;
}

void menu_vert::set_present_item ( int item1 )
{
 present_item=item1 ;
 if ( present_item > number_of_item-1 ) present_item = number_of_item-1 ;
}

void menu_vert::draw_items_cursor()
{
 int  i , j ;
  i = present_item ;
            textcolor(SELECTION_COLOR) ;
       textbackground(SELECTION_BK_COLOR) ;
          gotoxy(x+1,y+1+i) ;
          for ( j=x+1 ; i<=x+x_width  ; i++ )
          {
            gotoxy(j,y+1+i) ;
            cputs(" ") ;
          } ;
          gotoxy(x+1,y+1+i) ;
          cputs( item[i] ) ;
}

void menu_vert::hide_items_cursor()
{
 int  i , j ;
  i = present_item ;
          textcolor(FRAME_COLOR) ;
          textbackground(FRAME_BK_COLOR) ;
          gotoxy(x+1,y+1+i) ;
          for ( j=x+1 ; i<= x+x_width  ; i++ )
          {
            gotoxy(j,y+1+i) ;
            cputs(" ") ;
          } ;
          gotoxy(x+1,y+1+i) ;
          cputs( item[i] ) ;
}

main()
{
 menu_vert   abcd1 , abcd2 , abcd3  ;

  abcd1.init() ;

  abcd1.add_item("ЗАГРУЗИТЬ") ;
  abcd1.add_item("НОВЫЙ ФАЙЛ") ;
  abcd1.add_item("СОХРАНИТЬ") ;
  abcd1.add_item("ЗАПИСАТЬ В ...") ;
  abcd1.add_item("КАТАЛОГ") ;
  abcd1.add_item("ВЫХОД В DOS") ;
  abcd1.add_item("ПОКИНУТЬ - Alt X") ;
  abcd1.set_menu_xy(1,1) ;
  abcd1.draw_menu() ;
  abcd1.menu_choice() ;
  abcd1.erase_menu() ;

  abcd2.init() ;

  abcd2.add_item("КОНТРОЛЬ СКОБОК") ;
  abcd2.add_item("КОНТРОЛЬ СИНТАКСИСА") ;
  abcd2.set_menu_xy(20,1) ;
  abcd2.draw_menu() ;
  abcd2.menu_choice() ;
  abcd2.erase_menu() ;

  abcd3.init() ;

  abcd3.add_item("СПИСОК СИНТАКСИЧЕСКИХ ОШИБОК") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ ЛИТЕРАЛЫ") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ КОНСТАНТЫ") ;
  abcd3.add_item("ИСПОЛЬЗОВАННЫЕ СТАНДАРТНЫЕ ФУНКЦИИ") ;
  abcd3.add_item("ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ") ;
  abcd3.set_menu_xy(43,1) ;
  abcd3.draw_menu() ;
  abcd3.menu_choice() ;
  abcd3.erase_menu() ;

} // конец main
```

INCLUDE - файл KEYBOARD.H

```
#define    EASC_F1          1059
#define    EASC_F2          1060
#define    EASC_F3          1061
#define    EASC_F4          1062
#define    EASC_F5          1063
#define    EASC_F6          1064
#define    EASC_F7          1065
#define    EASC_F8          1066
#define    EASC_F9          1067
#define    EASC_F10         1068

#define    EASC_Shift_F1    1084
#define    EASC_Shift_F2    1085
#define    EASC_Shift_F3    1086
#define    EASC_Shift_F4    1087
#define    EASC_Shift_F5    1088
#define    EASC_Shift_F6    1089
#define    EASC_Shift_F7    1090
#define    EASC_Shift_F8    1091
#define    EASC_Shift_F9    1092
#define    EASC_Shift_F10   1093

#define    EASC_Ctrl_F1     1094
#define    EASC_Ctrl_F2     1095
#define    EASC_Ctrl_F3     1096
#define    EASC_Ctrl_F4     1097
#define    EASC_Ctrl_F5     1098
#define    EASC_Ctrl_F6     1099
#define    EASC_Ctrl_F7     1100
#define    EASC_Ctrl_F8     1101
#define    EASC_Ctrl_F9     1102
#define    EASC_Ctrl_F10    1103

#define    EASC_Alt_F1      1104
#define    EASC_Alt_F2      1105
#define    EASC_Alt_F3      1106
#define    EASC_Alt_F4      1107
#define    EASC_Alt_F5      1108
#define    EASC_Alt_F6      1109
#define    EASC_Alt_F7      1110
#define    EASC_Alt_F8      1111
#define    EASC_Alt_F9      1112
#define    EASC_Alt_F10     1113

#define    EASC_Home        1071
#define    EASC_Left        1075
#define    EASC_End         1079
#define    EASC_Up          1072
#define    EASC_Down        1080
#define    EASC_PgUp        1073
#define    EASC_Right       1077
#define    EASC_PgDn        1081

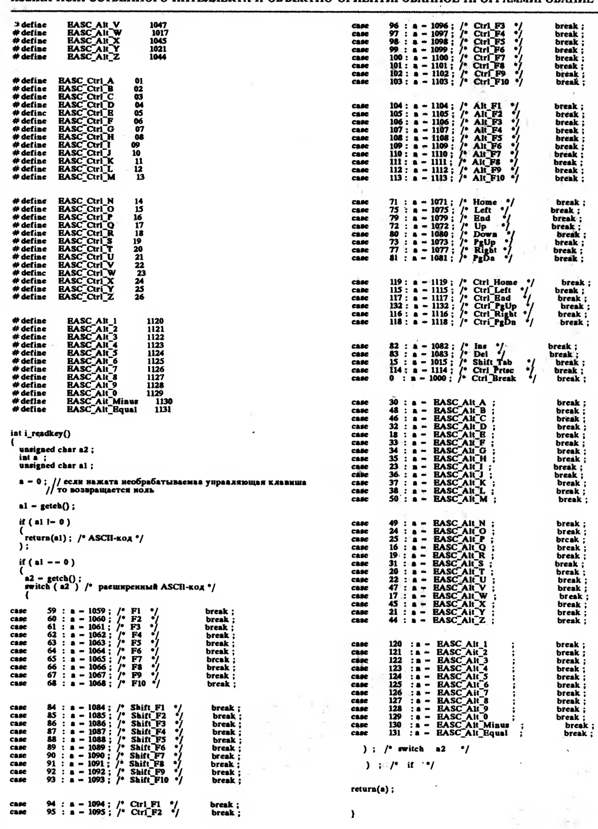
#define    EASC_Ctrl_Home   1119
#define    EASC_Ctrl_Left   1115
#define    EASC_Ctrl_End    1117
#define    EASC_Ctrl_PgUp   1132
#define    EASC_Ctrl_Right  1116
#define    EASC_Ctrl_PgDn   1118

#define    EASC_Ins         1082
#define    EASC_Del         1083
#define    EASC_Shift_Tab   1015
#define    EASC_Ctrl_Prtsc  1114
#define    EASC_Ctrl_Break  1000

#define    EASC_Enter       13
#define    EASC_Esc         27
#define    EASC_Space       32

#define    EASC_Alt_A       1030
#define    EASC_Alt_B       1048
#define    EASC_Alt_C       1046
#define    EASC_Alt_D       1032
#define    EASC_Alt_E       1018
#define    EASC_Alt_F       1033
#define    EASC_Alt_G       1034
#define    EASC_Alt_H       1035
#define    EASC_Alt_I       1023
#define    EASC_Alt_J       1036
#define    EASC_Alt_K       1037
#define    EASC_Alt_L       1038
#define    EASC_Alt_M       1050

#define    EASC_Alt_N       1049
#define    EASC_Alt_O       1024
#define    EASC_Alt_P       1025
#define    EASC_Alt_Q       1016
#define    EASC_Alt_R       1019
#define    EASC_Alt_S       1031
#define    EASC_Alt_T       1020
#define    EASC_Alt_U       1022
```



```
#define     EASC_Alt_V      1047
#define     EASC_Alt_W      1017
#define     EASC_Alt_X      1045
#define     EASC_Alt_Y      1021
#define     EASC_Alt_Z      1044

#define  EASC_Ctrl_A    01
#define  EASC_Ctrl_B    02
#define  EASC_Ctrl_C    03
#define  EASC_Ctrl_D    04
#define  EASC_Ctrl_E    05
#define  EASC_Ctrl_F    06
#define  EASC_Ctrl_G    07
#define  EASC_Ctrl_H    08
#define  EASC_Ctrl_I    09
#define  EASC_Ctrl_J    10
#define  EASC_Ctrl_K    11
#define  EASC_Ctrl_L    12
#define  EASC_Ctrl_M    13

#define  EASC_Ctrl_N    14
#define  EASC_Ctrl_O    15
#define  EASC_Ctrl_P    16
#define  EASC_Ctrl_Q    17
#define  EASC_Ctrl_R    18
#define  EASC_Ctrl_S    19
#define  EASC_Ctrl_T    20
#define  EASC_Ctrl_U    21
#define  EASC_Ctrl_V    22
#define  EASC_Ctrl_W    23
#define  EASC_Ctrl_X    24
#define  EASC_Ctrl_Y    25
#define  EASC_Ctrl_Z    26

#define     EASC_Alt_1      1120
#define     EASC_Alt_2      1121
#define     EASC_Alt_3      1122
#define     EASC_Alt_4      1123
#define     EASC_Alt_5      1124
#define     EASC_Alt_6      1125
#define     EASC_Alt_7      1126
#define     EASC_Alt_8      1127
#define     EASC_Alt_9      1128
#define     EASC_Alt_0      1129
#define     EASC_Alt_Minus    1130
#define     EASC_Alt_Equal    1131

int i_readkey()
{
  unsigned char a2 ;
  int a  ;
  unsigned char a1 ;

  a = 0 ; // если нажата необрабатываемая управляющая клавиша
          // то возвращается ноль

  a1 = getch() ;

  if ( a1 != 0 )
  {
   return(a1) ;  /* ASCII-код */
  } ;

  if ( a1 == 0 )
  {
   a2 = getch() ;
   switch ( a2 )  /*  расширенный ASCII-код */
   {
case     59  : a = 1059 ;  /*  F1   */          break ;
case     60  : a = 1060 ;  /*  F2   */          break ;
case     61  : a = 1061 ;  /*  F3   */          break ;
case     62  : a = 1062 ;  /*  F4   */          break ;
case     63  : a = 1063 ;  /*  F5   */          break ;
case     64  : a = 1064 ;  /*  F6   */          break ;
case     65  : a = 1065 ;  /*  F7   */          break ;
case     66  : a = 1066 ;  /*  F8   */          break ;
case     67  : a = 1067 ;  /*  F9   */          break ;
case     68  : a = 1068 ;  /*  F10  */          break ;

case     84  : a = 1084 ;  /*  Shift_F1   */    break ;
case     85  : a = 1085 ;  /*  Shift_F2   */    break ;
case     86  : a = 1086 ;  /*  Shift_F3   */    break ;
case     87  : a = 1087 ;  /*  Shift_F4   */    break ;
case     88  : a = 1088 ;  /*  Shift_F5   */    break ;
case     89  : a = 1089 ;  /*  Shift_F6   */    break ;
case     90  : a = 1090 ;  /*  Shift_F7   */    break ;
case     91  : a = 1091 ;  /*  Shift_F8   */    break ;
case     92  : a = 1092 ;  /*  Shift_F9   */    break ;
case     93  : a = 1093 ;  /*  Shift_F10  */    break ;

case     94  : a = 1094 ;  /*  Ctrl_F1   */     break ;
case     95  : a = 1095 ;  /*  Ctrl_F2   */     break ;
case     96  : a = 1096 ;  /*  Ctrl_F3   */     break ;
case     97  : a = 1097 ;  /*  Ctrl_F4   */     break ;
case     98  : a = 1098 ;  /*  Ctrl_F5   */     break ;
case     99  : a = 1099 ;  /*  Ctrl_F6   */     break ;
case     100 : a = 1100 ;  /*  Ctrl_F7   */     break ;
case     101 : a = 1101 ;  /*  Ctrl_F8   */     break ;
case     102 : a = 1102 ;  /*  Ctrl_F9   */     break ;
case     103 : a = 1103 ;  /*  Ctrl_F10  */     break ;

case     104 : a = 1104 ;  /*  Alt_F1   */      break ;
case     105 : a = 1105 ;  /*  Alt_F2   */      break ;
case     106 : a = 1106 ;  /*  Alt_F3   */      break ;
case     107 : a = 1107 ;  /*  Alt_F4   */      break ;
case     108 : a = 1108 ;  /*  Alt_F5   */      break ;
case     109 : a = 1109 ;  /*  Alt_F6   */      break ;
case     110 : a = 1110 ;  /*  Alt_F7   */      break ;
case     111 : a = 1111 ;  /*  Alt_F8   */      break ;
case     112 : a = 1112 ;  /*  Alt_F9   */      break ;
case     113 : a = 1113 ;  /*  Alt_F10  */      break ;

case     71  : a = 1071 ;  /*  Home   */        break ;
case     75  : a = 1075 ;  /*  Left   */        break ;
case     79  : a = 1079 ;  /*  End    */        break ;
case     72  : a = 1072 ;  /*  Up     */        break ;
case     80  : a = 1080 ;  /*  Down   */        break ;
case     73  : a = 1073 ;  /*  PgUp   */        break ;
case     77  : a = 1077 ;  /*  Right  */        break ;
case     81  : a = 1081 ;  /*  PgDn   */        break ;

case     119 : a = 1119 ;  /*  Ctrl_Home   */   break ;
case     115 : a = 1115 ;  /*  Ctrl_Left   */   break ;
case     117 : a = 1117 ;  /*  Ctrl_End    */   break ;
case     132 : a = 1132 ;  /*  Ctrl_PgUp   */   break ;
case     116 : a = 1116 ;  /*  Ctrl_Right  */   break ;
case     118 : a = 1118 ;  /*  Ctrl_PgDn   */   break ;

case     82  : a = 1082 ;  /*  Ins   */         break ;
case     83  : a = 1083 ;  /*  Del   */         break ;
case     15  : a = 1015 ;  /*  Shift_Tab   */   break ;
case     114 : a = 1114 ;  /*  Ctrl_Prtsc  */   break ;
case     0   : a = 1000 ;  /*  Ctrl_Break  */   break ;

case     30  : a = EASC_Alt_A ;                 break ;
case     48  : a = EASC_Alt_B ;                 break ;
case     46  : a = EASC_Alt_C ;                 break ;
case     32  : a = EASC_Alt_D ;                 break ;
case     18  : a = EASC_Alt_E ;                 break ;
case     33  : a = EASC_Alt_F ;                 break ;
case     34  : a = EASC_Alt_G ;                 break ;
case     35  : a = EASC_Alt_H ;                 break ;
case     23  : a = EASC_Alt_I ;                 break ;
case     36  : a = EASC_Alt_J ;                 break ;
case     37  : a = EASC_Alt_K ;                 break ;
case     38  : a = EASC_Alt_L ;                 break ;
case     50  : a = EASC_Alt_M ;                 break ;

case     49  : a = EASC_Alt_N ;                 break ;
case     24  : a = EASC_Alt_O ;                 break ;
case     25  : a = EASC_Alt_P ;                 break ;
case     16  : a = EASC_Alt_Q ;                 break ;
case     19  : a = EASC_Alt_R ;                 break ;
case     31  : a = EASC_Alt_S ;                 break ;
case     20  : a = EASC_Alt_T ;                 break ;
case     22  : a = EASC_Alt_U ;                 break ;
case     47  : a = EASC_Alt_V ;                 break ;
case     17  : a = EASC_Alt_W ;                 break ;
case     45  : a = EASC_Alt_X ;                 break ;
case     21  : a = EASC_Alt_Y ;                 break ;
case     44  : a = EASC_Alt_Z ;                 break ;

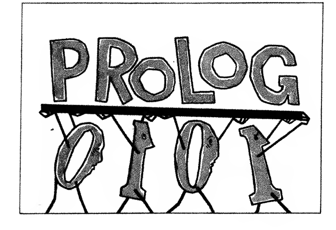
case     120  :a =  EASC_Alt_1      ;           break ;
case     121  :a =  EASC_Alt_2      ;           break ;
case     122  :a =  EASC_Alt_3      ;           break ;
case     123  :a =  EASC_Alt_4      ;           break ;
case     124  :a =  EASC_Alt_5      ;           break ;
case     125  :a =  EASC_Alt_6      ;           break ;
case     126  :a =  EASC_Alt_7      ;           break ;
case     127  :a =  EASC_Alt_8      ;           break ;
case     128  :a =  EASC_Alt_9      ;           break ;
case     129  :a =  EASC_Alt_0      ;           break ;
case     130  :a =  EASC_Alt_Minus   ;            break ;
case     131  :a =  EASC_Alt_Equal   ;            break ;

   } ;  /* switch   a2   */

  } ;  /*  if   */

return(a) ;

}
```



В основе языка логического программирования Пролог лежит логика предикатов первого порядка, точнее ограниченное подмножество логики предикатов, основанное на хорновских дизъюнктах [1]. Прежде чем рассмотреть логику предикатов, напомним некоторые положения логики высказываний (алгебры логики).

Математические основы языка Пролог

1. Логика высказываний

Логика высказываний — это раздел математической логики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логических операций над ними. Алгебра логики возникла в середине 19 века в трудах Дж.Буля.

Создание алгебры логики представляло собой попытку решать традиционные логические задачи алгебраическими методами. Основным предметом алгебры логики стали высказывания и логические операции над ними.

Высказывание — это повествовательное предложение, рассматриваемое вместе с его содержанием (смыслом) как истинное или ложное. Истину и ложь называют истинностными значениями высказываний. Иначе говоря, под высказываниями понимаются предложения, относительно которых имеет смысл утверждать, истинны они или ложны.

Примеры высказываний:

Следующие высказывания истинны:

1) Сегодня на улице тепло.
2) После окончания института студент получит диплом инженера.

Следующие высказывания ложны:

1) Диплом инженера студентам выдают на втором курсе.
2) Сегодня будет солнечное затмение.

Для обозначения истинности вводится символ "И", а для обозначения ложности — символ "Л". Вместо этих символов часто употребляются числа 1 и 0 .

С помощью пяти логических связок "и", "или", "эквивалентно", "если ..., то ..." и частицы "не" из уже заданных высказываний можно строить новые, более сложные высказывания.

Истинность или ложность получаемых таким образом сложных высказываний, зависит от истинности или ложности исходных высказываний и соответствующей трактовки связок.

Исчисление высказываний соединяет в себе сравнительную простоту с высокой содержательностью.

2. Логика предикатов первого порядка

В отличие от логики высказываний, логика предикатов первого порядка включает в себя термы, предикаты, переменные, кванторы. Это позволяет использовать ее для описания и решения задач практически в любых областях знаний.

Логика предикатов предназначена для описания логических законов, справедливых для любой непустой области объектов с произвольно заданными на этих объектах предикатами.

Элементарными компонентами логики предикатов являются предикатные символы, символы переменных, функциональные символы и символы констант.

Предикатный символ используется для представления отношений в некоторой области.

Запишем с помощью логики предикатов первого порядка следующее предложение:

— Все инженеры должны проходить обучение на факультете повышения квалификации.

Введем следующие обозначения:

A(x) — x является инженером;

B(x) — x должен проходить обучение на факультете повышения квалификации.

Используем квантор всеобщности, обозначающий "для всех", "для каждого", "для любого" и т.п. Запись (∀x) обозначает "для каждого x", "для любого x", "для всех x".

Тогда предложение будет иметь вид:

$$(\forall x)\ A(x) \rightarrow B(x)$$

т.е. "для всех x таких, что x является инженером, x должен проходить обучение на ФПК".

Аналогично может быть записано предложение: "Некоторые инженеры часто ездят в командировки".

Введем обозначение: C(x) — x часто ездит в командировки. Используем квантор существования, обозначающий "существует", "для некоторых". Запись (∃x) обозначает "существует x", "для некоторых x".

Тогда предложение будет иметь вид:

$$(\exists x)\ A(x) \rightarrow C(x)\ —$$

что означает "существуют такие x, что если x является инженером, то x часто ездит в командировки".

В логике предикатов формулировка теорем и аксиом развиваемой теории полностью записывается в виде формул, для чего употребляется особая символика, пользующаяся наряду с обычными математическими знаками и знаками для логических связок.

Всем логическим средствам, с помощью которых теоремы выводятся из аксиом, ставятся в соответствие правила вывода новых формул из уже выведенных. Эти правила формальны, т.е. таковы, что для проверки правильности их применений нет надобности вникать в смысл формул, к которым они применяются, и формулы, получаемой в результате; надо лишь убедиться, что эти формулы построены из определенного набора знаков, расположенных соответствующим образом.

Доказательство теоремы отображается в выводе выражающей ее формулы.

3. Логический вывод

Логический вывод можно рассматривать как последовательность формул, в конце которой и находится формула, подлежащая выводу. Формула, используемая при выводе, либо выражает аксиому, либо получается из одной или нескольких предыдущих формул по одному из правил вывода. Формула считается выводимой, если может быть построен ее вывод.

О доказуемости теорем в данной теории можно судить по выводимости выражающих их формул.

Выяснение выводимости или невыводимости той или иной формулы есть задача, не требующая привлечения далеко идущих абстракций, и решать эту задачу часто бывает возможно сравнительно элементарными методами.

Логический вывод — это формальный вывод в исчислении, содержащем логические правила и имеющем в качестве основных выводимых объектов формулы. Под логическим выводом понимают содержательное рассуждение, позволяющее от сформулированных аксиом и гипотез (допущений) переходить к новым утверждениям, логически вытекающим из исходных.

При фиксированных аксиомах и правилах логических переходов говорят, что последовательность формул является выводом некоторой формулы A из гипотез A1, ..., An ($n > 0$), если каждый член последовательности либо является аксиомой или одной из гипотез, либо получается из предыдущих формул последовательности по одному из заданных правил. При этом формула A называется выводимой из A1, ..., An.

Логическую программу можно использовать для поиска любой информации, которая логически следует из заданного в программе описания некоторой предметной области.

Для наглядного графического представления процесса логического вывода, а значит, и процесса решения задач, описываемых и решаемых с применением логических средств, целесообразно использовать некоторые понятия из теории графов.

Теория графов — раздел конечной математики, особенностью которого является геометрический подход к изучению объектов.

Граф задается множеством вершин (точек) и множеством ребер (связей), соединяющих некоторые (или все) пары вершин. При этом пары вершин могут соединяться несколькими ребрами.

Пример графа: Города Московской области и дороги, их соединяющие.

Определенные пары вершин графа соединены ребрами, и эти ребра направлены от одного элемента пары к другому. Граф, ребрам которого приписаны определенные направления, называется направленным.

Процесс логического вывода очень удобно изображать в виде направленного графа.

4. Стратегии поиска вершин графа

При выполнении логического вывода используется стратегия поиска в глубину. При таком поиске вершины списка просматриваются в порядке убывания их глубины в дереве поиска. Более глубокие вершины помещаются на первое место в списке для просмотра. Вершины, расположенные на одинаковой глубине, упорядочиваются слева направо.

Обычно при поиске используется вариант этой стратегии с возвращением. Возвращение проще реализуется и требует меньшего объема памяти. Стратегии с возвращением запоминают только один путь к целевой вершине; они не хранят полной записи процесса поиска, как это делают стратегии поиска на графе методом поиска в глубину.

Второй тип поиска называется поиском в ширину, поскольку просмотр вершин в дереве поиска происходит внутри одного "уровня", т.е. на одинаковой глубине. Поиск в ширину гарантирует нахождение кратчайшего пути к целевой вершине при условии, что такой путь вообще существует. Если такого пути нет, то данный метод закончит работу неудачно в случае конечных графов или никогда не закончит работу в случае бесконечных графов [1-2, 5-6].

Многие программы, созданные в области искусственного интеллекта, в значительной степени основаны на методах формальных логических рассуждений. Когда задача определяется в ограниченной области, подобные методы обеспечивают мощные средства для усечения дерева поиска решения [5-6].

5. Метод резолюции

Одним из наиболее популярных методов формальных рассуждений является метод логического вывода, использующий опровержение отрицания, т.е. метод доказательства "от противного", называемый методом резолюций.

Чтобы применить метод резолюций, сначала надо представить доказываемое утверждение с помощью логических формализмов исчисления предикатов. Затем следует отрицать доказываемое утверждение.

Далее метод резолюций применяется к набору аксиом, т.е. утверждений, заведомо справедливых в данной конкретной области или в данной ситуации, и к отрицанию доказываемого утверждения. Если в результате мы приходим к противоречию, т.е. в результате логического вывода получаем пустой дизъюнкт, обозначаемый □, то отрицание доказываемого утверждения должно быть ложным, а само утверждение — истинным.

При доказательстве по методу резолюций выполняется вычерчивание контрарных дизъюнктов.

Метод резолюций обладает свойством полноты, т.е. если исходное утверждение истинно, то в любом случае рано или поздно этот метод приводит к противоречию. Если же исходное утверждение ложно, то процесс вывода методом резолюций может оказаться бесконечным.

Однако метод резолюций имеет один существенный недостаток: число резолюций, выполняемых программой, растет экспоненциально как функция сложности задачи. Программы, успешно применяющие метод резолюций для решения небольших задач, как правило, не способны справиться с более сложными реальными задачами.

Лишь для малых задач можно создать программы, основанные только на методах формальных логических рассуждений.

Сила логических методов заключается в том, что они позволяют представить объекты и связи между ними в виде символов, которыми можно легко оперировать при помощи хорошо изученных методов, таких, как, например метод резолюций, осуществляя тем самым логические рассуждения.

Слабость логических методов заключается в том же, в чем и их сила — с помощью строгих логических формализмов невозможно представить большинство реальных задач, оперирующих неполными и неточными знаниями.

6. Эвристические правила

В настоящее время получили широкое распространение экспертные системы, в большинстве из которых используются правила типа "если ... то ..." ("если <условия> то <заключение>"), основанные на опыте экспертов, чьи знания заложены в экспертную систему. Такие правила называются эвристическими правилами, или эвристиками.

Каждое такое правило может быть простым само по себе. Однако с помощью системы таких правил, используемых как единое целое, оказывается возможным решать достаточно сложные задачи.

Программы, основанные на эвристических правилах, в отличие от программ, основанных на формально-логических методах, могут объяснять ход своих рассуждений в понятном для человека виде именно благодаря тому, что принимаемые программой решения основаны на правилах, полученных от экспертов, а не на абстрактных правилах формальной логики.

Формальные методы, даже если ими нельзя воспользоваться как основным средством рассуждения, могут сыграть полезную роль в управлении экспертными системами.

7. Комбинированные методы

Ранние программы в области искусственного интеллекта были основаны на каком-то одном подходе, чаще всего на формальных методах. Большое внимание было уделено созданию так называемого "универсального решателя задач", который был бы способен решать задачи в любой области знаний, основываясь на некоторых правилах формальной логики.

Сейчас многими специалистами признана целесообразность использования нескольких различных подходов [6].

Элементы знаний можно представить в программе разными способами. Каждый способ представления знаний может быть наиболее эффективен для одних операций и менее эффективен или вообще неэффективен для других.

Например, если программа должна эффективно отыскивать аналогии, то целесообразно представлять информацию о сравниваемых объектах в виде фреймов, состоящих из наборов ячеек, каждая из которых содержит значения атрибутов (свойств) объектов. Когда требуется найти аналогии между двумя объектами, представленными с помощью фреймов, то пустые ячейки одного фрейма заполняются значениями, взятыми из соответствующих ячеек другого.

Для всех систем логического программирования как в узком, так и в широком смысле, характерно одно

общее обстоятельство: для исполнения программ используются встроенные системы автоматического поиска вывода (или автоматического доказательства теорем).

Схема работы таких систем практически одинакова: на вход программы подается запрос вида "Найти X такие, что имеет место A(X) " (где A — формула, X — набор переменных). Далее система, используя заложенные в нее методы поиска вывода, пытается найти ответ на такой запрос. Поэтому методы поиска вывода, встроенные в систему, определяются логической программой и видом запросов.

8. Язык Пролог

В Прологе, в качестве формул, используются хорновские дизъюнкты. Большой класс практически важных задач допускает естественное, вполне приемлемое, описание с помощью этих формул. Кроме того, для них существвет эффективный метод поиска вывода.

Утверждения языка Пролог записываются в виде хорновских дизъюнктов, т.е. дизъюнктов, имеющих слева от знака ":-" не более одного литерала (атомарной формулы или ее отрицания).

Замечание. Здесь и далее используется синтаксис языка МПролог — одной из наиболее распространенных версий языка Пролог [2-4].

Утверждениям программы на языке Пролог соответствуют хорновские дизъюнкты с заголовками, т.е. дизъюнкты вида:

A:-B1,B2,...,Bn.

а целевому утверждению — дизъюнкт без заголовка:

:-B1,B2,...,Bn.

В программе может быть только один целевой дизъюнкт, в противном случае задача будет неразрешима.

Механизм поиска вывода, используемый в Прологе, берет свое начало от метода резолюций Робинсона. Формулами метода резолюций являются дизъюнкты, имеющие вид $A\,1 \vee \ldots \vee A\,n$, где $A\,1, \ldots, A\,n$ — литералы. Атомы называются положительными литералами, а их отрицания — отрицательными. Порядок литералов в дизъюнкте несуществен. Используемые в Прологе формулы A и $A\,1 \wedge \ldots \wedge A\,n \longrightarrow B$ эквивалентны хорновским дизъюнктам A и $\sim A\,1 \vee \ldots \vee \sim A\,n \vee B$, где знак "~" используется для обозначения отрицания.

Метод резолюций, получив на вход набор дизъюнктов, пытается построить вывод пустого дизъюнкта [] .

Текст утверждения языка Пролог "A:-B1,B2,...,Bn." может трактоваться двояко:

- как логическое утверждение, что A истинно, если одновременно истинны утверждения B1,B2, ... , Bk.
- либо как определение процедуры, утверждающее, что для того, чтобы выполнить процедуру A, надо выполнить все процедуры B1,B2,...,Bk. В сущности, обе интерпретации эквивалентны, что позволяет рассматривать язык Пролог и как алгоритмический язык программирования. Процедуры в правой части утверждеиия выполняются слева направо в порядке их записи.

Система Пролог основывается на процедуре доказательства теорем методом резолюций для хорновских дизъюнктов.

Система Пролог обладает встроенным механизмом логического вывода, благодаря чему, от пользователя требуется только описание своей задачи с помощью аппарата логики предикатов первого порядка, а поиск решения система берет на себя. Следует заметить, что для более эффективного использоваиия заложенного в язык Пролог механизма логического вывода при решении задачи, пользователь все-таки должен учитывать алгоритмические аспекты при описании своей задачи.

Вопросы:

1) Какой математический аппарат лежит в основе языка Пролог?

2) Что такое хорновские дизъюнкты?

3) Разрешима ли задача, не имеющая ни одного целевого дизъюнкта?

Ю.Тихонов

Литература:

1. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983.
2. Иванова Г.С., Тихонов Ю.В. Введение в язык МПролог. М.: Изд-во МГТУ, 1990.
3. Калиниченко Л.А., Степанов А.И., Тихонов Ю.В. Система МПролог для автоматизации обработки знаний на ЭВМ.//Сер. "Методические материалы и документация по пакетам прикладных программ". Вып. 59. М.: МЦНТИ, 1989.
4. Тихонов Ю.В. Язык логического программирования МПролог. КомпьютерПресс, № 5, 1991.
5. Нильсон Н. Принципы искусственного интеллекта. М.: Радио и связь, 1985.
6. Ленат Д.Б. Программное обеспечение систем искусствеиного интеллекта// В мире науки, 1984, № 11.

Фирма Mitsubishi Electric выпустила компьютер-записную книжку, содержащую в себе средства доступа к информационной системе MCA Service Information System, функционирующей по радио.

Эта машина является улучшенной версией нынешнего компьютера Maxy Note, к которому добавлены приемопередающее устройство, радиомодем, акустический соединитель и маленький принтер.

Весь набор без проблем умещается в чемоданчик размером 47x34x13 см. Он предназначен для очень деловых людей и стоит около 8800 долл. Аналогичная версия для хост-системы стоит 21500 долл.

Общая стоимость системы, может быть, высока, но передача данных в ней обходится дешевле, чем через обычные автомобильные телефоны. Кроме того, мобильные телефоны в Японии работают в основном внутри городской черты,а система MCA покрывает большую часть страны.

Newsbytes News Network, 19 July, 1991

Денежки счет любят...

Система Peachtree Complete III фирмы Peachtree Software

Система Peachtree Complete III представляет собой интегрированную бухгалтерскую систему, как и DacEasy Accounting, предназначенную для малых и средних предприятий. Наличие мощных средств работы с базой данных и модульность этого пакета делают его также приемлемым для отделений больших корпораций, имеющих свои бухгалтерские службы. В программу встроены мощные средства генерации отчетов, ревизорские и прочие бухгалтерские средства и включает следующие модули: "Гроссбух" (General Ledger), "Счета дебиторов" (Accounts Receivable), "Счета кредиторов" (Accounts Payable), "Фактурирование" (Invoicing), "Управление запасами" (Inventory), "Заказы на поставку" (Purchase Order), "Калькуляция себестоимости заказа" (Job Costing), "Основные средства" (Fixed Assets) и "Зарплата" (Payroll). Дополнительный модуль Peachtree Data Query III (PDQ) обеспечивает генерацию отчетов. Несмотря на то, что система представляет собой набор обособленных модулей, используемые в них файлы данных — счета, поставщики, потребители, продукция — общие и доступны всем модулям.

Как Peachtree Complete III, так и PDQ работают на компьютерах типа IBM PC и PS/2 и совместимых с ними, требуют для своего функционирования не менее 640 Кбайт оперативной памяти и 10 Мбайт памяти на жестком диске. Полностью установленная система, включая обучающий курс и справочную информацию, занимает около 8 Мбайт.

Система позволяет вести независимый бухгалтерский учет в подразделениях и дочерних компаниях, обеспечивая информацией бухгалтерию основной компании за счет соответствия планов счетов, — идентификация счетов отделов фирмы выполняется в последних двух разрядах кодов счетов. Работа каждого модуля характеризуется функциональной полнотой. Все операции пакета можно разбить на три группы: обработка, генерация отчетов и поддержка. Функции обработки позволяют выбрать и провести некоторую операцию. Функции генерации отчетов доступны в меню каждого модуля в виде перечня возможных сгенерированных отчетов. Функции поддержки позволяют изменять параметры настройки системы и добавлять записи в базу данных (например, новые виды продукции, новых поставщиков или покупателей).

Окончание. Начало в №№7—8.

Процесс инсталляции системы сопровождается вводом данных контрольного примера. В обучающем курсе показано, каким образом можно спроектировать и установить бухгалтерскую информацию собственной структуры, а также как выполнять различные операции и генерировать отчеты.

Собственно пользовательский интерфейс Peachtree Complete III выполнен в виде системы меню. Однако, в отличие от DacEasy Accounting, подменю главного меню представляются в виде всплывающих окон. Имеются средства для просмотра справочной информации: пользователю не требуется запоминать все коды, ему достаточно нажать при выполнении некоторой операции клавишу F2, и в окне будет высвечен соответствующий файл базы данных. Как и в DacEasy Accounting, здесь имеются средства для дополнения и корректировки базы данных. Навигация в меню Peachtree типична для большинства программ — с помощью курсора или нажатием первой буквы названия соответствующей команды. Если пользователь должен ввести некоторый параметр, то, как правило, система выводит на экран все его возможные значения.

Средства помощи контекстно-ориентированы, но, в отличие от большинства других программ, они не дают возможности просмотреть другую интересующую пользователя информацию за исключением относящейся к выполняемой в текущий момент функции.

Модуль "Гроссбух"

Модуль "Гроссбух" Peachtree Complete III работает в пакетном режиме. Модуль обрабатывает как непосредственно вводимую пользователем информацию, так и информацию, поступающую от других модулей системы. Пользователь также может задать повторяющиеся операции и операции, выполняемые в конце финансового периода, связанные с закрытием счетов и подготовкой гроссбуха к новому финансовому периоду. Модуль формирует итоги по каждому из счетов и распечатывает финансовые отчеты, которые могут, в

частности, содержать сравнеиие с аналогичными показателями выделеииого бюджета или показателями прошлых лет. Модуль может генерировать отчеты о результатах хозяйствеиной деятельиости подразделений.

Модуль "Гроссбух" Peachtree автоматически распределяет полученные доходы по нескольким (до пяти) счетам, что упрощает подсчеты в случае иаличия иескольких партнеров. Если в процессе иастройки Peachtree Complete III указывается, что корпорация включает несколько дочериих фирм (число которых не ограничено), то "Гроссбух" будет автоматически сводить итоги по всем подразделениям и генерировать сводиый финансовый отчет. Модуль позволяет как выводить иа экран, так и распечатывать финансовые отчеты. При выводе иа печать балаиса и финансового отчета оии могут одновремеино конкретизироваться по подразделеииям, кроме того, в отчетах пользователь может задать колонтитулы.

Модуль поддерживает трех- и четырехзиачную нумерацию счетов, воспринимая при этом соответствеиио 5 или 6 цифр (в последиих двух символах содержится код подразделения). Таким образом, пользователь может создать счета для 99 подразделений. Вместе с Peachtree поставляется плаи счетов, который пользователь может приият за осиову и модифицировать, оставить неизменным или полностью отказаться от него, создав собственный. План счетов подразделений должен точно соответствовать плану счетов компании, в противном случае будут получеиы некорректные результаты.

В поставляемый плаи входит 76 счетов, хотя их число может достигать 26000. Пользователь может как модифицировать, так и запретить модификацию плаиа счетов. В процессе модификации Peachtree Complete III обеспечивает мощный контроль правильности вводимой информации. Например, в конце модификации плаиа счетов от пользователя можио потребовать как указаиия общего числа измеиеиий, проведенных в плане счетов, так и контрольиого числа, полученного на основе иомеров счетов. Если эти значеиия ие совпадут с вычислеиными программой, то измеиения вводиться не будут. Подобиого рода строгий коитроль вполне оправдан, поскольку ошибка в плаие счетов приведет к сбою во всей дальнейшей работе системы.

Пользователь может просматривать план счетов как в целом, так и в выбраином диапазоне и, если потребуется, выводить иа экраи сальдо или содержимое счетов. На экраи могут выводиться начальное сальдо, конечиое сальдо и текущие операции.

Модуль "Гроссбух" имеет ряд средств, позволяющих адаптировать систему под конкретный вид деятельности. Пользователь может задать пароль, ограничивающий ввод определенных типов операций, как то корректировки счетов, начальиых сальдо и т.п. Пользователь может указать продолжительность храиеиия информации об операциях. Peachtree Complete III поддерживает 13 фииансовых периодов — 12 месяцев, а 13-й используется для проведения корректировок.

В рамках иитегрированиой системы "Гроссбух" воспринимает и обрабатывает ииформацию, поступающую от модулей — "Зарплата", "Счета кредиторов", "Счета дебиторов", "Основиые средства".

В регистре операций хранится список всех операций, проводившихся в течение месяца. При выводе этого отчета иа печать пользователь может отсортировать записи по номеру счета или коду источника. Описаиие каждой операции включает список всех задействованиых в ней счетов, дату, сумму, источник и некоторые другие параметры. Этот отчет используется при проведении ежемесячиой ревизии.

При проверке сбалаисированиости гроссбуха пользователь после проводки всех операций за месяц и передачи сводных журналов формирует промежуточный балаис. Балансовый отчет и отчет о результатах хозяйственной деятельиости может формироваться либо в стаидартной форме, либо в форме сравнения фактических данных с выделеиным бюджетом или финансовыми показателями предыдущих лет.

Модули "Счета дебиторов" и "Фактурирование"

Модуль "Счета дебиторов" обрабатывает операции, связаиные с покупателями, вычисляет величииы иалогов с продаж и платы за услуги, отслеживает неоплачеииые налоги и ведет базу даиных на 14400 покупателей. Счета поставщиков поддерживаются как по принципу открытого требования (open item), так и перенесеииого сальдо (balance forward). Кроме того, в системе могут обрабатываться и операции с непостоянными покупателями. В систему заносится достаточно большое количество информации о каждом покупателе, включая идентификациоиный код, назваиие, адрес, класс, тип счета, величину платы за услуги, код условий платежей и величину налога, при иеобходимости сообщение о наличии долга, предельную сумму кредита и адрес поставки. Средства поиска позволяют просматривать иифopмацию как о покупателях, так и об их счетах и объемах операций, проходящих на этих счетах.

Модуль "Фактурирование" использует те же данные о покупателях, что и "Счета дебиторов", и позволяет готовить счета-фактуры одиовременно с проводкой операций иа счетах. Модуль позволяет кодировать условия платежей и задавать до трех различных видов платы за услуги, которые могут вычисляться как процеит от средиего еженедельного или конечного сальдо иа счету покупателя. Модуль одиовременно с формированием счетов-фактур вносит требуемые измeнеиия в модули "Управление запасами" и "Счета дебиторов".

Модуль геиерирует два вида счетов-фактур — иа продукцию и услуги. В фактурах на продукцию указываются дата поставки, идеитификатор и описание изделия, цена за единицу и количество. Фактуры иа услуги имеют несколько более свободную форму, допуская включение в нее полного описания предоставлен-

ной услуги. Каждая запись в счете-фактуре не должна превышать 160 символов. С помощью поля "класс покупателя" можно указать, что выписка счета-фактуры должна осуществляться регулярно с некоторой периодичностью.

Наряду с фактурами модуль генерирует отчеты о работе с покупателями. Отчет может включать сообщение длиной до 75 символов, а также сообщение о необходимости срочного погашения долга. В отчет могут включаться операции, проводившиеся с покупателем по всем месяцам или только за текущий месяц, при этом для всего предшествующего финансового периода формируется лишь значение сальдо.

В совокупности оба модуля выполняют операции по проводке и распечатке счетов-фактур, выдаче кредитов, вычислению платы за обслуживание, проводке платежей, а также проводке счетов-фактур в гроссбухе и модуле "Управление запасами".

Средства просмотра файлов позволяют просматривать и модифицировать списки покупателей и продукции. Перед акцептованием любой из операций по желанию пользователя производится проверка на превышение предельной суммы покупки товара в кредит.

Система позволяет генерировать ряд отчетов, в том числе отчеты о налогах с оборота для налоговых управлений, отчеты о работе с покупателями, регистр всех операций по приходу средств за определенный период, регистр непроведенных счетов-фактур, регистр невыполненных заказов и отчет о задолженности покупателей.

При работе модулей совместно с модулем "Управление запасами" проводка платежей и возврата товаров автоматически изменяет величину запасов на складе. При работе модуля "Фактурирование" в сочетании с управлением запасами можно генерировать отчеты о недостающих, но требующихся товарах, а также отчеты о прибыльности различных видов товаров. Модуль "Счета дебиторов" также работает совместно с модулем "Калькуляция заказа" и проводит операции, связанные с выполнением заказов. Наконец, все операции, проводимые в модуле, отражаются в гроссбухе.

Модуль "Счета кредиторов"

Модуль "Счета кредиторов" регистрирует информацию о поставщиках, счетах-фактурах и кредитах; вычисляет потребность в наличности и генерирует отчеты о предоставлявшихся ранее скидках. Модуль проводит операции по закупке и оплате товаров и генерирует платежные поручения на оплату товара. В базе данных поставщиков может храниться до 14400 записей, кроме того, можно работать и со случайными поставщиками. Максимальная величина платежей не должна превышать 99.999.999 долларов 99 центов.

Модуль позволяет проводить фактурирование, кредитные операции и платежи. При выписке фактур на оплату модуль может руководствоваться заданным пользователем критерием, в частности, сроком погашения кредитного обязательства. После этого система будет автоматически генерировать платежные поручения, если пользователь в явном виде не укажет противное. Кроме того, перед печатью платежных поручений система выводит список всех подлежащих оплате счетов-фактур.

Система поддерживает до девяти регистров платежных документов банковских счетов и может автоматически выписывать платежные поручения в заданном пользователем формате. На корешке платежного поручения указываются дата, номер, сумма, наименование поставщика и адрес. Само платежное поручение содержит наименование и адрес поставщика, сумму прописью (естественно, на английском), защищенное поле с указанием суммы в долларах и центах, дату и номер платежного поручения.

Функции фактурирования позволяют осуществлять фиксированные платежи, как то проценты по ссудам или арендную плату, при этом пользователь должен указать номера месяцев, в которые производятся выплаты. Модуль автоматически формирует счета, указывая дату выплаты, сумму, дату, с которой начинается предоставление скидки, процент этой скидки. Такого рода выплаты могут осуществляться ежемесячно, раз в два месяца, раз в квартал, раз в полгода или раз в год. Как и в случае обыкновенных счетов-фактур, пользователь может перевести эти счета на любой из девяти банковских счетов, а сумму распределить между восемью счетами гроссбуха.

При просмотре файла поставщиков можно запросить информацию об их текущей деятельности, просроченных кредитах и неоплаченных счетах-фактурах, платежах и существующих ценах. В процессе проводки транзакций в специально открываемом окне можно просмотреть файлы поставщиков и продукции.

При генерации отчетов можно получить список открытых счетов-фактур, информацию о потребности в наличных деньгах на оплату товаров, приобретенных в кредит, по датам, регистр операций, в который включены все операции, проведенные в модуле "Счета кредиторов", ежемесячный регистр платежных поручений, список всех периодических платежей и список поставщиков. Модуль также позволяет проводить операции по выплате заработной платы сотрудникам, не работающим постоянно в фирме.

Модуль "Счета кредиторов" записывает информацию о проведенных операциях в гроссбух. Он также интегрирован с модулем "Заказы на поставку", который формирует заказы на поставку продукции поставщиками. Наконец, модуль работает совместно и с модулем "Калькуляция себестоимости заказа" при проводке операций, связанных с приобретением материалов для выполнения заказов.

Модуль "Заказы на поставку"

Модуль "Заказы на поставку" формирует заказы поставщикам на поставку продукции и отслеживает полные и частичные поставки. Кроме того, модуль взаимодействует с модулем "Управление запасами",

автоматически изменяя величину запаса при поставке. Модуль использует ту же базу поставщиков, что и "Счета кредиторов"; пользователи также могут просматривать и модифицировать списки поставщиков. То же относится и к базе данных запасов.

Заказы на поставку создаются с использованием стандартной информации, в них можно добавлять дополнительные данные. Допускается заказывать материалы, не регистрируемые в модуле "Управление запасами", а также изменять уже сформированные заказы. При частичных поставках заказ остается открытым вплоть до полной поставки или его закрытия. Однотипные изделия в поставке могут иметь разную цену, в этом случае каждому экземпляру присваивается уникальный номер и цена устанавливается по мере поступления продукции от поставщика.

Модуль позволяет печатать заказы на поставку изменения к заказам, и указания о прекращении поставки и генерировать списки планируемых поставок. Кроме того, в отдельный отчет помещаются невыполненные заказы с указанием поставщика и состояния поставки. Наконец, модуль генерирует список открытых заказов.

Рассматриваемый модуль является связующим между "Счетами кредиторов" и "Управлением запасами". Он модифицирует счета кредиторов и использует базу данных поставщиков. С другой стороны, при поступлении заказа этот модуль изменяет величину запасов и формирует справки для формирования последующих заказов.

Модуль "Управление запасами"

Модуль "Управление запасами" может хранить информацию о 19500 наименованиях изделий и имеет средства, позволяющие учитывать вхождение комплектующих элементов при сборке. Каждое изделие имеет 15-значный код, в котором указывается код типа изделия, код подразделения и номер экземпляра изделия. Кроме того, система отслеживает объем и цену поставки, объем полученной продукции и возврат, а также количество изделий, находящихся в незавершенном производстве (при сборке). Пользователь может включать в базу дополнительные описания для всех видов продукции, регистрируемых в модуле. Модуль хранит информацию о запасах на текущий и предыдущий месяцы или финансовые периоды, либо всю информацию текущего года.

Для определения себестоимости продукции может использоваться пять различных методов. Большинство операций, связанных с управлением запасами, могут проводиться из модулей "Фактурирование" и "Заказы на закупку". Основной задачей этого модуля являются в основном функции, связанные с ценообразованием, ведением информации о незавершенном производстве и выполнением операций по формированию отчетных документов.

Изменение величины себестоимости может производиться как автоматически, так и вручную. Пользователь может указать либо абсолютное значение, либо прирост в процентах.

Модуль позволяет генерировать ряд отчетов, в том числе список цен на хранящуюся продукцию с указанием названия, информации о поставщике, количества и трех уровней цен; отчет о состоянии запасов с указанием остатков на конец каждого месяца; отчеты по подразделениям; отчет по калькуляции себестоимости; отчеты о комплектации, содержащие список наименований компонентов, входящих в готовое изделие с указанием их количества; отчет о хранении запасов, позволяющий определять место физического расположения продукции на складах, и итоговый отчет, формируемый на конец месяца или года.

Модуль также генерирует регистр всех проводившихся операций, связанных с управлением запасами, включая и те, которые были сгенерированы из других модулей.

Модуль "Зарплата"

Модуль рассчитывает величину зарплаты, налогов и прочих удержаний и позволяет учитывать основные отпуска и отпуска по болезни. Максимальная численность, на которую может рассчитываться зарплата, составляет 3900 человек. Модуль также позволяет генерировать квитанции на оплату и проводить соответствующие операции, а также генерировать налоговые отчеты. Выплата зарплаты может проводиться раз в неделю, две недели, полмесяца или месяц. Модуль также имеет встроенную таблицу налогов, взимаемых в США по всем 50 штатам.

Пользователи могут создать таблицу для кассира, в которой указываются данные и зарплата каждого сотрудника. Также имеется программа для обработки исключительных ситуаций.

Модуль поддерживает несколько способов начисления зарплаты, включая любое сочетание окладной, почасовой зарплаты, разовых вознаграждений и комиссионных; вычисление доплат к зарплате за сверхурочную работу, работу во вторую и третью смены; выплату зарплаты в любом сочетании с недельным, двухнедельным, полумесячным и месячным циклом; все виды налогов, принятых в США; три дополнительных вида доходов и шесть дополнительных видов скидок; автоматический учет дополнительных отпусков за сверхурочно отработанное время.

Модуль проводит все операции в гроссбух, с разнесением окладов по трем разным журналам и зарплаты рабочих-почасовиков по десяти различным счетам.

Модуль ведет базу данных сотрудников, в которую включаются код, имя и адрес сотрудника, величина доходов и другая информация.

После ввода величины зарплаты система вычисляет сумму к выдаче и генерирует чек на получение денег, а также отчеты о величине зарплаты и удержаний.

Модуль взаимодействует с модулями "Гроссбух" при проводке операций начисления зарплаты и "Калькуля-

ция себестоимости заказа", который позволяет автоматически определять зарплату за выполнение определенных работ по заказам.

Модуль "Калькуляция себестоимости заказа"

Модуль "Калькуляция себестоимости заказа" позволяет определять затраты на выполнение заказа. В число затрат включаются расходы на материалы, оплату рабочей силы и накладные расходы. Модуль также вычисляет величину доходов от выполнения заказа. Допускается разбивать выполнение заказа на этапы и вводить информацию о завершении этапа и его оплате, сроках начала выполнения заказа, проценте выполнения, а также сопоставлять предварительные и фактические величины затрат. В зависимости от величины фактической себестоимости допускается изменение цены.

После начала выполнения заказа он заносится в файл заказов, где содержится информация об этапе, на котором он находится, необходимых для его выполнения производственных мощностях, материальных и трудовых затратах. Код затрат включает информацию о количестве рабочих часов, необходимых для выполнения заказа, и код материальных ресурсов, база данных которых хранится в модуле "Управление запасами". Модуль также использует файл покупателей, поддерживаемый модулем "Счета дебиторов", и взаимодействует с модулем "Зарплата", внося изменения в зарплату сотрудников, выполняющих заказы.

В процессе работы пользователь может получить справку о состоянии выполнения заказов, где отражаются наименование заказчика, количество изменений к заказу, процент выполнения, исходная цена, цена с учетом корректировок и себестоимость заказа. Модуль позволяет получать отчеты, содержащие данные о величине рентабельности заказа и себестоимости по статьям калькуляции.

Модуль "Основные средства"

Модуль "Основные средства" ведет информацию о производственных и конторских зданиях, сооружениях, оборудовании и позволяет рассчитывать величину износа и амортизации, а также хранить в базе данных до 12000 записей, добавлять и удалять (списывать) новые записи, а также изменять величину износа.

Модуль поддерживает 13 различных методов определения износа, а также может переходить при его определении от одного метода к другому, например, из соображений снижения налогов.

Модуль позволяет генерировать ряд отчетов, в том числе содержащих информацию о приобретении, ликвидации, износе основных средств; графики износа основных средств за неделю, месяц, квартал, полугодие и год.

При выполнении операций модуля "Основные средства" автоматически модифицируются соответствующие журналы гроссбуха.

Дополнительные возможности

Все модули PeachTree Complete III обеспечивают защиту от несанкционированного доступа. При доступе к данным имеется два уровня парольной защиты. Первый уровень позволяет также устанавливать сами пароли.

Система, кроме того, имеет полуавтоматические средства восстановления данных. При изменении файлов она напоминает пользователю о необходимости создания резервных копий. Ответ "yes" на подсказку приведет к записи на диск измененных файлов.

Генерация отчетов

Каждый модуль PeachTree Complete поддерживает достаточное число отчетов, начиная с распечаток соответствующих файлов, и кончая сложными бухгалтерскими отчетами по специфике модуля. Пользователь имеет определенную степень свободы в выборе формата выводимого на печать отчета, например, платежного поручения или заказа на закупку.

Наконец, к PeachTree Complete поставляется дополнительный модуль PeachTree Data Query III (PDQ), обеспечивающий доступ пользователю к базе данных и генерацию отчетов. С помощью PDQ пользователь может извлекать значения полей из любых файлов базы данных, формировать списки, графики и нестандартные отчеты. Модуль обеспечивает экспорт в форматы Lotus 1-2-3 и Symphony, Microsoft Multiplan и Visicalc, Ashton-Tate dBASE, а также в виде файлов ASCII и DIF.

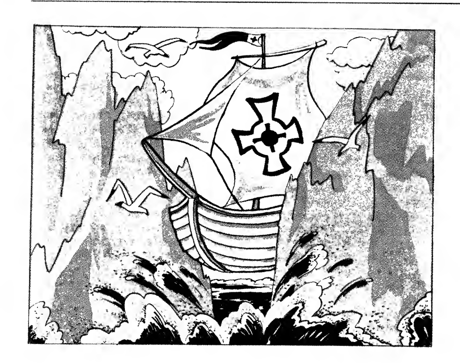
Модуль имеет меню-ориентированный интерфейс и позволяет отбирать и обрабатывать поля информации, получая промежуточные и окончательные итоги, выполнять над полями арифметические и логические операции, сортировать их и получать средние значения.

В целом программа PeachTree Complete III выполняет все основные бухгалтерские процедуры и может полностью удовлетворить потребности малых и средних предприятий. Безусловным достоинством системы является наличие модуля "Основные средства", а также мощных возможностей работы с базой данных и генерации отчетов. Наконец, его цена на фоне достаточно сильных возможностей выглядит чрезвычайно низкой — всего 200 долларов.

Однако наряду с достоинствами, программа имеет такие недостатки, как отсутствие сетевых средств и обучающей программы.

М.Михайлов

По материалам:
Faulkner Technical Reports on Microcomputers and Software;
DataPro Reports on Microcomputers.



Между прочим...

ДВЕ ОПЕРАЦИОННЫХ СИСТЕМЫ НА ДИСКЕ

Некоторые виды программного обеспечения и специфические платы требуют использования конкретной версии операционной системы (обычно MS-DOS 4.0). В то же время ваше программное обеспечение может быть оптимизировано под использование той версии DOS, которая установлена. Что делать в таком случае?

Самый простой и надежный путь — записать в различные каталоги (или на разные диски) разные версии операционных систем. Но здесь нужно учитывать, что MS-DOS умеет грузиться только с дисков A: и C:. Поэтому основную систему имеет смысл установить на диск C:, а вторую загружать с гибкого диска.

При этом нужно иметь отдельный каталог для каждой системы и при загрузке подключать лишь один из них — это необходимо для того, чтобы все утилиты работали как следует.

Не забудьте в файле AUTOEXEC.BAT описать переменную COMSPEC так, чтобы система искала командный процессор (COMMAND.COM) на диске C:, а не на A:, например так (если командный процессор находится в корневом каталоге):

```
SET COMSPEC = C:\COMMAND.COM
```

В противном случае вам придется постоянно менять гибкие диски, вставляя системный взамен рабочего.

Этот метод замедляет загрузку, но с этим вполне можно мириться.

Более хитрый метод заключается в использовании специальных пакетов для разметки жесткого диска. Обычно они позволяют выбрать раздел диска, с которого будет производиться загрузка. Однако этот путь можно рекомендовать только наиболее опытным и аккуратным пользователям, ведь одно неверное движение — и ваш жесткий диск прикажет долго жить, притом восстановить его будет не слишком просто.

Так что лучше пользоваться более простым методом, описанным выше.

КАК ИСПОЛЬЗОВАТЬ ДИСКИ ТИПОВ, ОТСУТСТВУЮЩИХ В ПЗУ BIOS

Нередко случается, что купленный винчестер не соответствует ни одному из типов накопителей, присутствующих в постоянном запоминающем устройстве, содержащем BIOS. При этом возникают всяческие

проблемы — в лучшем случае просто не используется часть диска, в худшем — не удается отформатировать его вообще.

Эту проблему можно обойти как минимум тремя путями. Первый состоит в использовании утилиты "наращивания" BIOS, содержащей информацию об очень многих типах накопителей самых разных изготовителей и записывающий ее в КМОП-ОЗУ вашего компьютера (то есть туда, где обычно хранится информация о его конфигурации). Таких утилит довольно много, поэтому используйте ту, которая будет содержать описание винчестера той марки, которая вас интересует. В качестве примеров таких утилит можно привести Vfeature Delux фирмы Golden Bow Systems, California Ten Pack фирмы California Software Product.

Второй путь — использование специализированных пакетов форматирования дисков Disk Manager и SpeedStor. Оба включают в себя длинный список накопителей самых различных фирм и, кроме того, позволяют вручную ввести параметры накопителя, если он не соответствует ни одному из предлагаемых типов. В сущности, этот режим аналогичен использованию описанных выше утилит — при этом точно так же сведения о диске записываются в подпитываемое ОЗУ.

Третий путь проще двух первых, но не всегда он является наилучшим. Можно просто сменить BIOS на более свежую версию. Для этого нужно знать его название и фирму-изготовителя (чаще всего используется BIOS одной из трех марок: Phoenix, Award или AMI). Узнать ее элементарно — как правило, при загрузке системы первым сообщением является сообщение именно о марке BIOS'a. Если вы не увидите его, придется снять крышку компьютера и прочитать наклейку на микросхеме с BIOS. Обратите внимание на номер его версии и дату создания. Имея эти сведения, можно поискать более новую версию BIOS, но перед покупкой стоит выяснить, есть ли в ней интересующий вас тип накопителя.

Первый метод самый дешевый — обычно такая утилита стоит совсем недорого, второй — самый дорогой, но он дает вам дополнительные возможности по подготовке накопителей на жестких дисках к работе. Замена BIOS занимает промежуточное положение по цене, но может преподнести несколько приятных сюрпризов — например, исчезнет ошибка, с которой вы боролись все время работы на данном компьютере, или добавится поддержка более современных устройств.

ОСВОБОЖДЕНИЕ МЕСТА ПОД РИСУНКИ ПРИ РАБОТЕ С MICROSOFT WORD

Если вы готовите какую-либо публикацию (например, документацию или бюллетень), которая будет размножаться с помощью ротапринта или копировальной машины, вам может понадобиться вставить в оригинал-макет фотографии или что-нибудь, нарисованное от руки. Конечно, можно отсканировать изображение и вверстать его на компьютере. Но для того, чтобы потом получить приличное качество фотографии, качества обычного лазерного принтера будет недостаточно. Поэтому-то столь живуч дедовский метод "ножниц и клея".

Итак, чтобы оставить место под рисунок, следует воспользоваться опцией Format pOsition.

Прежде всего нужно создать новый параграф (абзац) — то-есть просто нажать на Enter. Далее с помощью этой команды нужно сформатировать данный абзац требуемым образом. При этом вы можете задать размер оставляемого чистым "окошка", его положение на листе, можете привязать его к конкретному месту на полосе.

Процедура заключается в следующем. Создав абзац, поместите на него курсор и перейдите в меню Format pOsition. Измерьте ширину вашей иллюстрации и внесите ее в графу frame width. Ширину полей вокруг картинки внесите в графу distance from text. Положение окошка на полосе определяется графами horisontal

frame position и vertical frame position, а также дополняющими их графами relative to. Вы можете выбрать некоторое стандартное положение — например, в центре полосы (обе графы описаны как centered). Кроме того, можно задать положение в абсолютных единицах — то есть отступы слева и сверху. Стандартным является положение абзаца In Line — в том месте, где он был вставлен в текст. В этом случае он привязывается к конкретному месту в тексте.

Несколько слов о значении графы relative to. Что такое column, думаю, понятно всем. Margins обозначает, что все расстояния будут считаться не от края листа, а от полей, описанных в Format Division Margins. Выбор режима page позволяет сместить блок за пределы текстовой полосы. Если абзац, к тому же, обведен рамкой, можно получить весьма интересный эффект. Здесь можно также использовать абсолютные величины сдвигов, очень точно определяя положение блока на полосе.

Ну и последняя операция — установка высоты блока. Для этого сохраните все изменения, внесенные в пункт Format pOsition, и перейдите к пункту Format Paragraph. Здесь задайте нужную высоту блока, изменив интерлиньяж абзаца (line spasing). Замечу, что тут можно использовать любые принятые в MS Word единицы длины, но обязательно нужно указать, какие единицы вы имеете в виду (по умолчанию считается, что вы задали размер в линиях — что-то около 4.236 мм).

Место под иллюстрацию можно отбить линейками, либо заключить в рамку. Делается это с помощью пункта меню Format Border.

НЕКОРРЕКТНАЯ ОБРАБОТКА ФУНКЦИИ ACHOICE() БИБЛИОТЕКИ EXTEND.LIB КОМПИЛЯТОРА CLIPPER SUMMER'87

Всем хороша функция ACHOICE() компилятора Clipper Summer'87 фирмы Nantucket. Однако при написании контекстного HELPa для АРМа, где эта функция широко применялась, я столкнулся с некоторой некорректностью в ее работе, заключающейся в том, что после вызова клавишей F1 процедуры подсказки из меню, созданного этой функцией, передвижения по альтернативам меню вызывают искажения экрана, а именно: справа от того места, где была выведена подсказка, появляется текст альтернативы выбора меню (этот текст должен был быть выведен там, где стоял подкрашенный выделенным (enhanced) фоном маркер (prompt).

Ниже приведен текст демонстрационной программы. Откомпилируйте, слинкуйте и стартуйте этот пример. При появлении меню нажмите клавишу F1 (появится подсказка), затем любую другую клавишу и снова "погуляйте" по меню. Вы увидите, что текущая альтернатива восстановится не на своем месте. Видимо, функция ACHOICE() не запоминает положение курсора при нажатии клавиши F1 и после отработки процедуры HELP выводит на экран свое текущее сообщение уже не на то место, куда следовало бы.

Способ борьбы с этим, который я считаю недостойным проколом фирмы Nantucket, состоит в следующем. При входе в процедуру HELP запомните текущее положение курсора, а при выходе — восстановите. Соответствующие операторы демонстрационного примера закрыты комментариями, откройте их (строки 12,13 и 20) и ACHOICE() не исказит картинку после выхода пользователя из HELPa.

Кстати, в документации фирмы утверждается, что при вызове процедуры HELP, переменная CALL_PRG содержит имя вызывающей процедуры. Однако в нашем случае эта переменная будет содержать не строку символов "DEMO", а строку "ACHOICE()" (предлагаю убедиться в этом самостоятельно), что несколько усложняет написание процедуры справки в части блокировки рекурсивных вызовов. Можно предположить, что ACHOICE() написана на языке dBASE (Clipper), хотя и в этом случае стоило бы позаботиться о корректном содержимом переменной CALL_PRG.

Текст пакетного файла трансляции, редактирования и выполнения:

```
clipper demo
tlink demo,demo,demo,clipper extend
demo
```

P.S. Вышеописанные неточности исправлены в Clipper *5*.0.

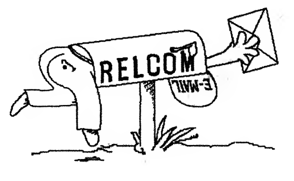
Файл demo.prg:

```
declare m[5]
@ 0,0 clear to 24,79
afill (m, 'альтернатива')
set color to w/b
@ 1,1,7,16 box space(9)
@ 1,3 say 'ВЫБЕРИТЕ ##'
@ 7,4 say 'F1 — HELP'
achoice(2,2,6,15,m)
@ 0,0 clear to 24,79
*
procedure HELP
****     k = col()
****     l = row()
save screen to scr
oldcolor = setcolor()
set color to n/bg
@ 13,18 say 'Нажмите любую клавишу и погуляйте по меню'
inkey(0)
setcolor(oldcolor)
****     @ l,k say''
restore screen from scr
```

И.Вязаничев
В.Макаренков

По материалам:
PC/Computing, LAN Times

demos/*



- Сегодня еще можно включиться в национальную и в мировую систему электронной почты, став пользователем сети RELCOM **
- Демос/* обеспечит подключение к сети, а так же, при необходимости, поставит оборудование: компьютер, телефонный модем и программное обеспечение

Оборудование фирмы HP	Компьютеры, лазерные принтеры, плотеры и другая периферия фирмы Hewlett-Packard реализуется со скидкой. При покупке лазерного принтера LaserJet III за валюту - скидка до 31%!!! Гарантийное обслуживание 3 года.
Модемы MNP-5 **снижены цены**	2400/4800 bps встроенные и внешние, адаптированные к отечественным линиям, эффективно работающие в почтовой сети. Коррекция ошибок, компрессия данных, Hayes совместимые, аттестованы Минсвязи СССР. Гарантийное обслуживание 1 год.
Компьютерные сети **снижены цены**	Локальные и глобальные. Работы по установке и наладке. Документация на русском языке по ОС Novell NetWare. Подключение локальной сети к электронной почте RELCOM.
Издательские системы	Шрифтовые кассеты кириллицы для лазерных принтеров Canon, HP LaserJet. Загружаемые шрифты кириллицы для лазерных принтеров Canon, LaserJet и моделей, совместимых с ними. Микропрограммы кириллицы: прошивка ПЗУ принтеров, адаптеров мониторов и пр.
Оригинальное SoftWare	ОС ДЕМОС 2.2 для СМ-1700, СМ-4, Электроника-85/79. Прикладные программы для систем, совместимых с ОС UNIX. Пакеты русификации систем SCO Xenix, MS Windows.
Системы Автоматизации	Платы (IBM PC AT/XT) АЦП-ЦАП. Блок АЦП: 20 разрядов, связь по RS-232 (заказ). Платы релейных коммутаторов и цифровых каналов. Платы цифровых каналов (до 24 вх/вых.). Платы интерфейса канала общего пользования. Платы ЦАП. Платы аналоговых усилителей. Контроллер крейта КАМАК для IBM PC AT/XT.

ДЕМОС/*: 113035 Москва, Овчинниковская наб. дом 6/1, телефон: 231-21-29, 231-63-95; Fax: 233-5016; E-mail: info@hq.demos.su

** Электронная почта сети Relcom создана Демос/* и ИВЦ ИАЭ им. Курчатова, и зарегистрирована Международным центром в Стенфорде (США). По вопросам заключения договоров на подключение к сети Relcom обращаться по телефонам Демоса/* и телефону ИВЦ: 196-72-50.



НОВОСТИ

Два директора фирмы Sun Microsystems посетили Москву с очень интересным визитом. Они, видимо, являются одними из последних представителей фирм, предлагающих советским компьютерным командам участвовать в совместных научных разработках.

Дэйвид Дитзел, директор по перспективным разработкам, и Джордж Тэйлор, директор по экспериментальным архитектурам, прочитали лекцию на факультете вычислительной математики и кибернетики МГУ, осветив технические возможности Sparc-технологии.

Фирма Sun Microsystems, вероятно, будет сотрудничать с лучшими разработчиками ИС в СССР — это команда из Института точной механики и вычислительной техники.

Компания Sun хочет влить свежую кровь в свои инженерные команды и находит, что специалисты из ИТМиВТ своим умом дошли до ряда моментов, использованных в создании Sparc-процессоров.

Sun имеет в СССР эксклюзивного дистрибьютора — фирму Просистем (как отмечают некоторые эксперты, это не самый лучший выбор).

Newsbytes News Network, 15 June, 1991

Фирма IBM закончила первый квартал 1991 года со следующими финансовыми показателями:
- объем продаж аппаратного обеспечения снизился на 17%;
- прибыли от продажи сетевого оборудования упали на 48%;
- продажа оборудования в США шла особенно плохо; в Европе — хорошо;
- объем продаж программного обеспечения возрос на 13%; его сумма составила 2.35 миллиарда долларов;
- объем услуг возрос 18,9%;
- лизинг возрос на 16,2%.

IBM Computer Today, 24 Apr.-7 May, 1991

Фирма IBM USSR открыла свой первый учебный центр в Ленинграде. Этот центр, который будет функционировать при помощи имеющихся в Ленинграде пяти бизнес-партнеров IBM, предназначен для обучения пользователей работе с IBM-компьютерами.

Фирма переехала с Покровского бульвара в большее помещение на улице Веснина. За новое помещение Моссовет получает 500 тысяч долларов в год. В старом помещении открывается еще один учебный центр — для обучения продавцов IBM-техники.

Newsbytes News Network, 11 July 1991

СП "ПараГраф" начало продажу русификатора базы данных Paradox. Об этом было объявлено на форуме мира ПК в Москве.

Русификатор, который поставляется в виде отдельной программы, прилагаемой к фирменной коробке с последней (3.5) версией Paradox, полностью убирает английские буквы с экрана монитора. Все элементы меню, подсказки весьма аккуратно переведены на рус-

ский язык. Кроме того, с русификатором можно задавать русские имена полей в базе данных и производить сортировки по алфавиту — оригинальный продукт этого делать с русскими буквами не мог.

Русификация, произведенная ПараГрафом самостоятельно, была полностью одобрена фирмой Borland.

Это уже не первая версия русификатора. Первая была создана более года назад. Эти работы ведутся группой программистов под руководством Павла Зелинского.

Вместе с коробкой Paradox и полностью переведенной русской документацией все удовольствие стоит 9900 рублей. Хотите купить — звоните в ПараГраф (095) 200-25-66.

Newsbytes News Network, 12 July 1991

Как изменились ограничения КОКОМ на экспорт в Советский Союз??

Можно ввозить любые персональные компьютеры с процессором 80386. 80486 все еще находится в списке товаров, требующих получения лицензии на ввоз. IBM System/36, MicroVAX 3 и другие подобные машины теперь также можно купить без лицензии.

Вместо старого ограничения на винчестеры — 150 Мбайт емкости и скорость передачи 10 Мбит/с — осталось лишь одно — скорость передачи информации от привода не должна превосходить 25 Мбит/с.

Любое программное обеспечение к компьютерам, которые можно легально ввозить, лицензирования не требует.

По-прежнему под запретом — локальные сети на более чем 200 пользователей, оборудование для волоконно-оптических линий связи, программы, обеспечивающие гибкую маршрутизацию сообщений в компьютерных сетях.

Newsbytes News Network, 14 July 1991

Палата представителей конгресса США приняла законопроект о перераспределении частот. Палата представителей проголосовала за законопроект, согласно которому полоса электромагнитного спектра шириной 200 МГц, ранее использовавшаяся военными, передается для использования в гражданских целях. Но будущее законопроекта остается туманным из-за споров с Белым домом о способе распределения этих новых частот. Белый дом хочет, чтобы право пользования этими частотами было вынесено на аукцион. Конгресс же считает, что их надо передать бесплатно после слушаний, в ходе которых определится, какие предложения наилучшим образом отвечают общественным интересам.

Teleputing Hotline / NewsBytes

Фирма GTE создает совместное предприятие с Советским Союзом. Фирма GTE создаст 240 телефонных линий, которые свяжут Москву с остальным миром с помощью микроволновой и спутниковой связи. Проект будет выполняться вместе с находящейся в США фирмой San-Francisco — Moscow Teleport и советским Министерством связи. По заявлению GTE, СП Sovintel начнет предоставлять услуги в ноябре за твердую валюту для гостиниц и деловых центров в Москве. В настоящее время тем, кто хочет осуществить звонок, приходится несколько часов ждать, пока линия не освободится. Московские специалисты считают, однако, что новое предприятие вряд ли разрушит монополию СП Comstar, в настоящее время единственного предприятия, предоставляющего услуги по установлению быстрой международной телефонной связи местным деловым людям. На данный момент Министерство связи утвердило создание более 150 совместных предприятий.

Teleputing Hotline / NewsBytes

Цифровой переносной телефон: японская новинка. NEC объявила о создании 195-граммового цифрового сотового переносного телефона. Он немного больше аналогичного изделия "Mova" фирмы NTT, но меньше, чем Microtac фирмы Motorola. В нем используются схемы фирмы NTT, которая предоставляет свою технологию также фирмам Fujitsu, Matsushita, Mitsubishi и Motorola. Ожидается, что работа телефонной переносной цифровой сотовой службы начнется в Токио осенью 1992 года. В отличие от ситуации в США, цифровые телефоны в Японии будут несовместимы с аналоговыми.

Teleputing Hotline / NewsBytes

Toshiba выпускает новые изделия в Японии и США. Фирма Toshiba выпустила на рынок продукт, который она называет самой быстрой в мире переносной рабочей станцией (класс laptop). SPARC LT/AS1000 использует RISC процессор фирмы Sun Microsystems и работает со скоростью 17.5 миллионов операций в секунду (MIPS) — прошлогодняя модель работала со скоростью 13.4 MIPS. AS1000 имеет встроенный жесткий диск на 329 Мбайт, вместо

которого можно установить диск емкостью 3 Гбайта.

Фирма Sun будет продавать эти станции под своим именем. Среди покупателей — Fujitsu, Oki, Unisys Japan и Fuji Xerox. Цена — 1.78 миллионов йен (13000 долларов).

В США Toshiba выпустила в продажу модем T24D/X. Он работает на скорости 9,600 бод для переносных PC на сотовой и проводной телефонной сети. Он совместим со старыми компьютерами, начиная с T1200.

Toshiba USA также выпустила три новых компьютера — записных книжки (notebooks) серии 2000. Они весят 6.9 фунта (2.9 кг.), и в двух из них используется процессор Intel 80386SX.

Федеральная торговая комиссия (FTC) сочла японские компании виновными в поставке плоских экранов на американский рынок по демпинговым ценам и может поднять таможенные тарифы на дисплеи на жидких кристаллах до 63%. Американские компьютерные компании выступают против этих тарифов, заявляя, что они увеличат стоимость их продукции, а это заставит их уступить рынок японским конкурентам.

Teleputing Hotline / NewsBytes

Неполадки в телефонной сети США. Ошибки в программном обеспечении неизбежны, и это надо учитывать. Для создания цифровой телефонной сети нужно заменить все аппаратные соединения на программные. А в программном обеспечении по самой его природе обязательно будут ошибки.

Такие ошибки могут дорого обойтись, как выяснила компания DSC Communications. Похоже, что выпущенная в апреле очередная версия программного обеспечения для их Signal Transfer Points (Точек передачи сигнала) не прошла тщательной проверки. Программное обеспечение отказало, и STP прекратили работу, когда произошел перепад в подаче электроэнергии в Балтиморе. Это повторилось в Лос-Анджелесе, потом в Питтсбурге. Каждый раз при этом миллионы пользователей оставались без телефонной связи.

STP являются ключевыми компонентами цифровой сети. В сети ISDN все сигналы, определяющие маршруты звонков, отделяются от самих звонков и проходят через STP. Если STP прекращают работу, становится невозможно позвонить. А STP работают на программном обеспечении.

Обнаруженная проблема будет решена с помощью дополнительной программы, которая будет действовать как предохранитель. Перепады электроэнергии не будут выводить из строя все устройство целиком, а только его часть. И эти сбои, в свою очередь, не приведут к отключению коммутаторов, потом коммутационных станций, а затем целых городов.

Но что произойдет в следующий раз? Большинство представителей компаний согласны с тем, что нужны дублирующие пути про запас. Если одна дорога завалена, вы идете по другой. Большие компании подстрахуются, построив собственные сети или используя так называемые "обходные" компании. Но, если верить Федеральной комиссии по связи (FCC), остальным может и не повезти.

Есть и другая проблема, специфическая для фирмы DSC. Существует вероятность того, что фирма понесет юридическую ответственность с тяжелыми финансовыми последствиями. Вся компания в настоящее время поставлена под удар. Если бы такая неисправность случилась в станциях фирм Northern Telecom, AT&T или Ericsson, такой угрозы бы не существовало.

Teleputing Hotline / NewsBytes

AT&T не сочувствует ранам, нанесенным хакерами фирме Mitsubishi. Mitsubishi не получила поддержки ни от AT&T, ни от Федеральной комиссии по связи США по вопросу возбуждения дела, направленного на возвращение фирме денег, украденных хакерами (компьютерными взломщиками) через PBX (местную АТС фирмы) в Нью-Йорке. Mitsubishi требовала от AT&T возмещения 10.4 миллионов долларов плюс оплаты судебных издержек за звонки, сделанные компьютерными преступниками, начиная с 1988 года через System 85 — системы обмена частными сообщениями в филиале фирмы (PBX).

Наилучшие PBX такие, как System 85, позволяют всем работникам компании звонить друг другу — даже с других континентов — с помощью 6-значного дополнительного кода, что делает эти системы уязвимыми. Хакеры звонили по бесплатному номеру, использовавшемуся для набора этого шестизначного дополнительного кода, а затем разговаривали со всем миром бесплатно. Mitsubishi, однако, не хочет брать на себя ответственность, заявляя, что PBX произведена AT&T и, стало быть, та во всем виновата.

Недавно в FCC поступила жалоба компании Chartways, которая пыталась возложить на AT&T оплату междугородных звонков в похожей ситуации. Но FCC в мае не только отвергла жалобу, но вынесла решение, что "владелец PBX обладает лучшими возможностями по поддержанию безопасности системы, и он

несет ответственность", если безопасность нарушена. AT&T заявила, что компании известно о проблемах с безопасностью, и у нее есть необходимое программное обеспечение, позволяющее от них избавиться. Но за использованием PBX должно вестись регулярное наблюдение. И если вы не выполняете рекомендованные действия, то вам может крупно не повезти.

Teleputing Hotline / NewsBytes

Стандарту по безопасности брошен вызов. США сдерживают принятие стандарта по безопасности для передачи компьютерных сообщений, настаивая на том, что Агентство Национальной Безопасности должно всегда иметь возможность расшифровать сообщения, используя стандарт. Агентство борется за требование использовать алгоритм, названный ElGamal, запатентованный в США, в то время как в компьютерной индустрии более популярна схема шифрации RSA. RSA негодует, но правительство заявило, что ElGamal позволит сэкономить деньги. Похоже, что единственный коммерческий продукт, основанный на схеме шифрации ElGamal, — это программа Secret Agency фирмы Information Security из Деерфилда, Иллинойс.

Teleputing Hotline / NewsBytes

Смоленск: местные власти поднимают телефонные тарифы. Власти Смоленска решили поднять тарифы на телефонные разговоры по собственной инициативе. Такое происходит в Советском Союзе впервые. Ранее на услуги телефонной связи тарифы устанавливались правительством из Центра. Плата за установку телефона увеличилась на 50%, фиксированная месячная плата за пользование удвоилась. Для предприятий тарифы увеличились в 10 раз. "Радио России" подвергло решение критике.

Teleputing Hotline / NewsBytes

SuperTablet с вводом с клавиатуры/электронным карандашом. Фирма Tusk из Флориды сообщила, что ею разработан компьютер-планшет All-Terrain SuperTablet, позволяющий осуществлять ввод как с помощью электронного карандаша, так и с клавиатуры. Он будет, начиная с IV квартала этого года, соперничать с производимым фирмой NCR компьютером 3125, использующим электронный карандаш. Кроме клавиатуры, компьютер, по заявлению компании, отличается упрочненным дизайном, который может выдержать удар пули, но весит всего 6 фунтов (2.5 кг.) против 3.9 фунта (1.6 кг.), которые весит компьютер NCR. Оба компьютера могут работать с MS-DOS, OS/2, PenApps, Penpoint, Windows 3.0 и Unix. Компьютер фирмы Tusk будет стоить 6000 долларов, на 1,2000 долларов больше, чем компьютер NCR.

Teleputing Hotline / NewsBytes

COPIA INTERNATIONAL анонсировала программу Credit Card Per Fax, которая позволит бюро с факсовыми службами получить номер кредитной карточки, зарегистрировать покупку в банковском учреждении и переслать платежный документ. Компания заявила, что вся процедура занимает обычно около минуты.

Teleputing Hotline / NewsBytes

MITSUBISHI объявила о создании опытной версии новой факс-машины, которая пересылает страницу информации по телефонным линиям за 5 секунд, в три раза быстрее действующих моделей. В ней используются сжатие данных и более скоростные модемы, и она могла бы стоить намного меньше, чем существующие факс-машины серии Group IV.

ФИРМА ROHM из Японии утроит производство дисплеев на жидких кристаллах, используемых в компьютерах класса laptop. Фирма предсказывает, что в скором времени они будут использоваться в системах навигации автомобилей.

Teleputing Hotline / NewsBytes

СП Интермикро стало первым официальным представителем фирмы Apple в Советском Союзе.

Созданное в 1988 году совместное советско-австрийское предприятие будет выполнять работы по локализации программного обеспечения для Макинтоша и созданию сети дилеров по стране.

Сделку финансирует финансовая группа Prosystem, владеющая 60 % уставного капитала СП Интермикро.

По утверждению ряда экспертов, это уже не первая попытка фирмы Apple выйти на советский рынок.

В СССР, по заявлению представителей компании, будут свободно продаваться все модели Макинтоша. Возможно, в будущем новые модели компьютеров на процессоре Motorola 68040 потребуют получения лицензии на ввоз.

Поставки компьютеров первым советским покупателям начнутся в середине сентября этого года. Примерно в ноябре появится полностью переведенная на русский язык документация к компьютеру. Эту работу заканчивает группа сотрудников МГУ, в которую входят не только программисты, но и, что очень важно, филологи и лингвисты. Тестовая версия операционной системы, по заявлению Интермикро, существует уже и сейчас.

Компьютеры будут продаваться по ценам, близким к европейским.

Технический директор СП Интермикро Анатолий Карачинский сказал также, что они ожидают потока русифицированных программ для Макинтоша в течение одного-полутора лет.

Уже есть русские версии редактора текстов, и издательского пакета Quark Xpress 3.1. Версия Ventura Publisher появится в течение полугода.

Как сообщил в начале месяца представитель фирмы Microsoft Дейл Кристиансен, их фирма, которая выпускает и программы для Мака, еще не рассматривала вопрос об их локализации. "Мы сможем быстро сделать это сразу после того, как получим от Apple версию русской операционной системы".

Newsbytes News Network,
23 July, 1991

Фирма Sharp начинает поставки в СССР нескольких новых типов портативных компьютеров. Теперь кроме PC-6220 можно купить еще три типа машин класса notebook и лаптоп с цветным монитором.

PC-6240 — почти аналогична PC-6220. Отличие заключается в емкости винчестера: если у 6220 она составляет 20 Мбайт, то у 6240 — 40. Остальный параметры остались теми же: процессор 80С286, тактовая частота 12/7.16/6 МГц, ОЗУ 1 Мбайт с возможностью расширения до 3 Мбайт, гнездо для сопроцессора 80С287 (12 МГц), время доступа к диску 23 мс, монитор фирмы Sharp соответствует стандарту VGA и обеспечивает разрешение 640x480 точек и 16 оттенков серого. Вес компьютера 2 кг, толщина всего 34 мм, время работы от батарей несколько уменьшилось — 1.7 ч. против 2 ч. PC-6240, как и 6220, снабжен операционной системой и пакетом LapLink, размещенными в ПЗУ.

PC-6521 и PC-6541 открывают новую серию notebook-компьютеров PC-6500. В ней использован тот же монитор, что и в серии PC-6200. Этот компьютер также построен на базе процессора 80С286 12/7.16/6 МГц с возможностью использования сопроцессора. ОЗУ емкостью 1 Мбайт расширяется до 4 Мбайт с помощью модулей в 1 и 2 Мбайта. Жесткий диск имеет емкость 20 Мбайт (PC-6521) и 40 Мбайт (PC-6541) при времени доступа 23 мс. В отличие от 6200, машины серии 6500 оборудованы встроенным накопителем на гибких дисках 3.5" 1.44 Мбайта, но и весят больше — 2.9 кг. Внешне эта машина имеет много общего с TravelMate 3000.

И, наконец, очередной шедевр фирмы, которая сейчас является лидером в области разработки и производства жидкокристаллических мониторов — лаптоп PC-8501 с цветным экраном. Это мощный и вполне

транспортабельный компьютер, построенный на процессоре 80386DX с тактовой частотой 20/8 МГц. Можно использовать сопроцессор 80387DX с тактовой 20 МГц.

Монитор соответствует стандарту VGA, обеспечивая воспроизведение 16 цветов при разрешении 640x480 точек и 256 цветов при разрешении 320x200 или 360x480 точек. Дисплей с диагональю 10.4 дюйма создан с использованием технологии TFT — тонкопленочных транзисторов, которая позволяет получить изображение с почти фотографическим качеством. Компьютер имеет ряд необычных дополнительных режимов работы, повышающих качество воспроизведения цветных изображений. Эта машина (как, впрочим и другие портативные компьютеры Sharp с цветным экраном) разрабатывалась в основном для проведения презентаций. Поэтому вполне логичным является то, что изображение может дублироваться на внешнем мониторе, в том числе и на проекционном.

Жесткий диск имеет емкость 100 Мбайт, накопитель для гибких дисков 3.5 дюйма 1.44 Мбайта также не забыт. Оперативная память — 2 Мбайта. Возможно расширение до 10 Мбайт.

Весит эта машина 6.9 кг, питается только от сети. Она довольно громоздка (318x399x94 мм) и напоминает первые лаптопы, но это — совершенно другой уровень технологии. Дополнительно фирма предлагает внешний накопитель для 5.25-дюймовых дисков, такой же, как и для PC-6500.

Заказ

Советско-американское предприятие "Соваминко"
Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 191186, Ленинград, Невский проспект, 28, Магазин № 1 "Дом книги"

От кого . . . .

Адрес . . . .
(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

Заказ

Советско-американское предприятие "Соваминко"
Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 630076, Новосибирск, Красный проспект, 60 Магазин № 7 "Техническая книга"
Телефон для справок: 20-05-09

От кого . . . .

Адрес . . . .
(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .


HEWLETT PACKARD
ARUS
ПРЕДЛАГАЕТ
ЗА РУБЛИ, ЗА ДОЛЛАРЫ
ПОСТАВКА СО СКЛАДОВ В МОСКВЕ
Высокоэффективный лазерный принтер Hewlett Packard LaserJet IIIP, технические характеристики которого приближаются к характеристикам широко известного принтера LaserJet III, а цена значительно ниже.
Кроме того, мы поставляем разнообразную вычислительную технику, периферийное оборудование, принтеры, плоттеры, сканеры и расходные материалы производства фирмы Hewlett Packard.
Продажа через дилерскую сеть фирмы ARUS Handels A.G.
тел. (095)230-5612
факс. (095)230-2182
Малое предприятие "КиМ"
тел. (095)220-3185
факс. (095)230-2182
IMPACT
1
2
3
4


Объявлена
подписка на
журнал КомпьютерПресс
1992 г.
тел. 420.83.80.
491.01.53.
Dr. Dobb's
UNIX

Цена 3.15